MuZero with Self-competition for Rate Control in VP9 Video Compression

Amol Mandhane[*†] **Anton Zhernov**[*†] **Maribeth Rauh**[*†] **Chenjie Gu**[*†] **Miaosen Wang**[†]

Flora Xue[†] **Wendy Shang**[†] **Derek Pang**[‡] **Rene Claus**[‡] **Ching-Han Chiang**[‡]

Cheng Chen[‡] **Jingning Han**[‡] **Angie Chen**[†] **Daniel J. Mankowitz**[†] **Jackson Broshear**[†]

Julian Schrittwieser[†] **Thomas Hubert**[†] **Oriol Vinyals**[†] **Timothy Mann**[†]

Abstract

Video streaming usage has seen a significant rise as entertainment, education, and business increasingly rely on online video. Optimizing video compression has the potential to increase access and quality of content to users, and reduce energy use and costs overall. In this paper, we present an application of the MuZero algorithm to the challenge of video compression. Specifically, we target the problem of learning a rate control policy to select the quantization parameters (QP) in the encoding process of libvpx, an open source VP9 video compression library widely used by popular video-on-demand (VOD) services. We treat this as a sequential decision making problem to maximize the video quality with an episodic constraint imposed by the target bitrate. Notably, we introduce a novel self-competition based reward mechanism to solve constrained RL with variable constraint satisfaction difficulty, which is challenging for existing constrained RL methods. We demonstrate that the MuZero-based rate control achieves an average 6.28% reduction in size of the compressed videos for the same delivered video quality level (measured as PSNR BD-rate) compared to libvpx's two-pass VBR rate control policy, while having better constraint satisfaction behavior.

1 Introduction

In recent years, video traffic has dominated global internet traffic and is expected to grow further [Cisco, 2020]. *Efficient video encoding algorithms* are key to reducing bandwidth and storage costs, as well as improving user Quality of Experience (QoE) in scenarios such as video-on-demand (VOD) and live streaming.

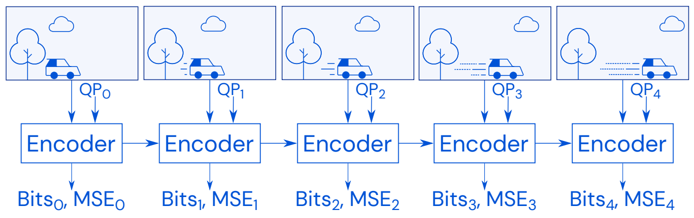
In video compression, rate control is a critical component that determines the trade-off between rate (video size) and distortion (video quality) by assigning a quantization parameter (QP) for each frame in the video. A lower QP results in lower distortion (higher quality), but uses more bits. In this paper, we focus on the rate control problem in libvpx [WebM, 2010b], an open source VP9 codec library

[*]Equal contributions

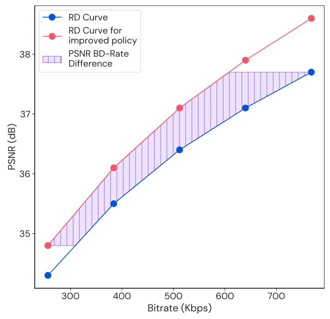
[†]DeepMind, London, UK

[‡]Google, Mountain View, CA, USA

Correspondence to: Amol Mandhane <mandhane@deepmind.com>, Anton Zhernov <typedef@deepmind.com>, Chenjie Gu <gcj@deepmind.com>



(a) Overview of the Rate Control process. For illustration purpose, we only draw the show frames. In practice, the encoder also assigns a QP for hidden alternate reference frames (ARF).



(b) Example of Rate-Distortion (RD) curve.

Figure 1: Illustrative example of the rate control process, Rate-Distortion curve, and the BD-rate metric for encoding efficiency.

[Mukherjee et al., 2013]. The objective in this setting is to select a QP for each of the video frames to maximize the quality of the encoded video subject to a bitrate constraint as seen in (1).

$$\max_{\text{QPs}} \text{ Encoded Video Quality} \quad \text{s.t. Bitrate} \leq \text{Target} \tag{1}$$

The rate control problem can be seen as a constrained planning problem. In particular, the QPs are chosen sequentially for each video frame as seen in Figure 1a, with the goal of optimizing for video quality under a bitrate constraint. Classical optimization algorithms such as mixed integer linear programming cannot be used for rate control as the codec operates as a black box without an explicit equation and the interactions between the video frames cannot easily be modeled by a linear objective. To provide some intuition for the planning element, imagine a video that has little motion for the first half of the video (e.g., a person talking or a static scene), and then has a lot of motion for the latter half of the video (e.g., a sports highlight). In this case, using up more of the bitrate budget on the first half of the video will cause the second half of the video to have lower quality. Intuitively, the rate control algorithm needs to allocate more bits to complex/dynamic frames and less bits to simple/static frames. This requires the algorithm to understand the rate-distortion tradeoff for each frame and their relationship to the QP values. However, the inter-dependency of QP decisions and the rate-distortion metrics are often non-trivial, making it challenging to engineer optimal rate control algorithms that work for a diverse set of videos.

Reinforcement Learning (RL) [Sutton and Barto, 2018] is a sequential decision making framework that has had recent success in solving numerous planning problems (in many cases achieving super-human performance) ranging from games [Mnih et al., 2013, Tessler et al., 2017, Vinyals et al., 2019, Silver et al., 2016] to chip design [Mirhoseini et al., 2020] and robotics [Levine et al., 2016]. RL algorithms are now solving problems at scale and are potentially well-suited for solving rate control.

In this work, we apply the MuZero RL algorithm [Schrittwieser et al., 2020] along with a novel self-competition based constrained RL method to rate control in VP9. Specifically, our main contributions are as follows:

- We propose a self-competition based reward mechanism that optimizes for the constrained rate control objective.
- We combine this self-competition based reward mechanism with the MuZero algorithm to create 'MuZero-RC', an RL based rate control agent.
- We evaluate this agent against libvpx's two-pass VBR rate control implementation [WebM, 2017] on 3062 video clips, each of which are 5-seconds in length, obtained from the YouTube UGC dataset [Wang et al., 2019] which has been created for evaluating video compression algorithms. We show that MuZero-RC achieves an average 6.28% reduction in bitrate (measured as PSNR BD-rate) compared to the libvpx baseline which is canonically used for VP9 encoding, while having better bitrate constraint satisfaction.

2 Related Work

Rate control in video codecs: Traditionally, rate control algorithms are based on empirical models using relevant features, such as mean absolute differences (MAD) and sum of absolute transformed differences (SATD) from past encoded frames. Common empirical models include quadratic models used in MPEG-4's VM8 [Tihao Chiang and Ya-Qin Zhang, 1996] and H.264/AVC [Ma et al., 2005, Minqiang Jiang and Nam Ling, 2005, Kwon et al., 2007], ρ-domain models that map between rate and the proportion of non-zero quantized coefficients [He and Mitra, 2002], and dynamic programming based models that exploit interframe dependency [Jiangtao Wen et al., 2000]. In modern codecs, such as HEVC and VVC, Li et al. [2014] introduced λ-domain model to control both rate and mode selections through rate-distortion optimization instead of relying on QP decisions. Despite recent advances in rate control models, the empirical rate control models require complex hand-designed heuristics to effectively adapt to different dynamic video sequences and satisfy different requirements for different applications.

ML for rate control: Machine learning enables more complex and nonlinear models to be formulated. Sanz-Rodriguez and Diaz-de-Maria [2011] use a radial basis function network to predict QP offset and ensure quality consistency. Gao et al. [2016] utilize a game theory method to allocate CTU-level bit allocation and optimize for SSIM in HEVC. More recently, Reinforcement learning approaches in rate control have also been proposed to HEVC [Hu et al., 2018, Zhou et al., 2020, Kwong et al., 2020, Ho et al., 2021, Chen et al., 2018]. Mao et al. [2020] use an imitation learning approach on evolutionary search based policy [Salimans et al., 2017] with a feedback-based correction for rate control in VP9.

Unlike the previously proposed ML based approaches, our proposed solution can work with a wide range of target bitrates specified for longer video durations with multiple groups-of-pictures (GOPs) instead of specifying bitrates for each GOP, which is close to the practical setting. The solution does not rely on demonstrations from prior experience, which allows discovery of rate control strategies from scratch.

3 Background

3.1 VP9 Video Compression

In this section, we briefly describe the two-pass variable bitrate (VBR) mode, which is a popular libvpx-vp9 rate control mode for Videos-On Demand (VOD). In this work, we use the statistics collected in the first-pass and the second-pass encoding as the input features to a neural network model that predicts the QP for every frame.

First-pass encoding: The encoder computes statistics for every frame in the video, by dividing the frame into non-overlapping 16×16 blocks followed by per-block intra and inter-frame prediction (of each block) and calculation of the residual error. The statistics contain information such as average motion prediction error, average intra-frame prediction error, average motion vector magnitude, percentage of zero motion blocks, noise energy, etc. [WebM, 2010a].

Second-pass encoding: The encoder uses the first-pass statistics to decide key-frame locations and insert hidden alternate reference frames. The key-frames and alternate reference frames are used as references for encoding other frames, so their encoding quality affects other frames in the video as well. With those decisions made, the encoder starts to encode video frames *sequentially* as seen in Figure 1a. The rate controller regulates the trade-off between rate and distortion by specifying a QP to each frame in order to maximize the quality and reduce the bits as explained in section 4.1. The QP is an integer in range [0, 255] that can be monotonically mapped to a quantization step size which is used for quantizing the frequency transform of the prediction residue for entropy coding. Smaller quantization step sizes lead to smaller reconstruction error (measured by mean squared error), but also higher bits usage for the frame.

3.2 Constrained Markov Decision Process

A Markov Decision Process (MDP) [Sutton and Barto, 2018] $\mathcal{M}$ is defined by the tuple $\langle S, A, P, R, \gamma \rangle$ where S is the set of states; A is the set of available state-dependent actions; $R : S \times A \to \mathbb{R}$ is a bounded reward function; $P : S \times A \times S \to [0, 1]$ is the transition function, where $P(s'|s, a)$ is the probability of transitioning to state s' from s when action a was taken; $\gamma \in [0,1]$ is the discount factor. A policy $\pi : S \times A \to [0,1]$ is a mapping from a state to a distribution over actions where $\pi(a|s)$ denotes the probability of taking action a in state s. The goal is to obtain a policy which maximizes the expected sum of discounted rewards (J^R_π).

$$\max_\pi J^R_\pi(s) := \mathop{\mathbb{E}}_{a\sim\pi, s\sim P}\left[\sum_t \gamma^t R(s_t, a_t)\right]$$

A Constrained Markov Decision Process (CMDP) [Altman, 1999] extends the MDP formulation by introducing k constraint functions $c_k : S \times A \to \mathbb{R}$ and corresponding thresholds β_k. The goal is to obtain a policy which maximizes the expected sum of discounted rewards subject to the discounted constraints ($J^{c_k}_\pi$) being under the target thresholds β_k.

$$\max_\pi J^R_\pi \quad \text{s.t.} \quad J^{c_k}_\pi \le \beta_k \; \forall k \tag{2}$$

There are a number of approaches to solving this constrained RL objective [Altman, 1999, Tessler et al., 2018, Efroni et al., 2020, Achiam et al., 2017, Bohez et al., 2019, Chow et al., 2018, Paternain et al., 2019, Zhang et al., 2020]. One approach involves solving the Lagrangian relaxation of the original problem as $\max_\pi \min_{\lambda \ge 0} J^R_\pi - \lambda(J^{c_k}_\pi - \beta_k)$. Here, the typical approach is to alternate the optimization between the policy π and the Lagrangian parameter λ [Tessler et al., 2018, Calian et al., 2021]. Another powerful approach is to fix the Lagrangian parameter and perform reward shaping [Achiam et al., 2017, Tessler et al., 2018].

3.3 MuZero

MuZero [Schrittwieser et al., 2020] is a Reinforcement Learning algorithm based on deep neural networks and Monte-Carlo tree search (MCTS) planning algorithm [Coulom, 2006] which has achieved state-of-the-art performance in various domains like Chess, Go, Shogi, and Atari. MuZero uses a learned model of the environment dynamics with MCTS to look ahead and plan from a given state in order to identify good actions. It uses a policy neural network as a prior for the MCTS, and a value network to truncate the MCTS. Unlike its predecessors, MuZero does not need to interact with the environment for planning; instead it uses a learned model of environment dynamics to look ahead.

4 Rate Control with MuZero

The following section is divided into three parts. First, we formulate the rate control problem as a CMDP and define the rate control CMDP objective that we wish to optimize. Next, we introduce the self-competition based reward mechanism to address this constrained objective that can be incorporated into any RL agent. Finally, we provide an overview of the architecture and training setup of our MuZero agent combined with the self-competition based reward for rate control (MuZero-RC).

4.1 Rate Control as a Constrained RL Problem

In this paper, we focus on the 'two-pass, variable bitrate (VBR)' mode in the libvpx implementation of VP9. However, our methods do not focus on the specifics of this mode and can be generalized to other settings. The objective of rate control in this mode is to maximize the quality of the encoded video while keeping the size under a user-specified target. Quality of the encoding is commonly measured as Peak Signal-to-Noise Ratio (PSNR) $:= 10\log_{10}\left(255^2/\text{MSE(original, encoded)}\right)$, although any other quality measure such as SSIM [Wang et al., 2004] or VMAF [Li et al., 2016] can be used.

The rate control problem can be formulated as the Constrained MDP (CMDP) tuple $\langle S, A, P, R_{\text{PSNR}}, \gamma = 1, c_{\text{Bitrate}}, \beta_{\text{Target}}\rangle$. The state space S consists of the first-pass statistics and the frame information generated by the encoder (described in section 4.3). The action space A consists of an integer QP in the range $[0, 255]$ which is applied to the frame being encoded. The transition function $P(s'|s,a)$ transitions to a new state $s' \in S$ (next frame to be encoded) given that action $a \in A$ (QP) was executed from state s (current frame being encoded) as seen in Figure 1a. The bounded reward function R_{PSNR} is defined as the PSNR of the encoded video at the final timestep of the episode and 0 otherwise. The constraint function c_{Bitrate} is the bitrate of the encoded video at the final timestep of the episode and 0 otherwise. β_{Target} is the user-specified size target for the encoding. We consider the size target range of [256, 768] kbps (kilobits-per-second) for 480p videos in this paper, although our methods can be extended to larger ranges. The rate control constrained objective to optimize is defined as:

$$\max_{\pi(\text{QP}_t|s_t,\beta_{\text{target}})} J_\pi^{\text{PSNR}} \quad \text{s.t. } J_\pi^{\text{Bitrate}} \leq \beta_{\text{Target}} \tag{3}$$

As mentioned in Section 3.2, it is possible to solve the unconstrained Lagrangian relaxation of the CMDP objective by performing alternating optimization on the Lagrange parameter λ and the policy π. However, as mentioned in previous works (e.g., [Tessler et al., 2018]), multi-timescale stochastic approximation is very sensitive to learning rates and is therefore typically difficult to tune. Additionally, for learning a rate control policy, a large number of Lagrangian parameters would be needed to solve the constrained RL objective (per video and per target bitrate) as there is no single multiplier that is optimal across videos and bitrates. We validated this by combining MuZero with the Lagrangian relaxation method and observed that it was both difficult to tune during training and was unable to provide a reasonable solution in terms of quality and constraint satisfaction during evaluation. A more in-depth analysis can be found in the appendix A.4. As such, we propose a different approach which we refer to as self-competition which solves a slightly different form of the above CMDP objective and removes the need to use Lagrangian parameters.

4.2 Constrained RL via Self-Competition

In this section, we present the self-competition reward mechanism that enables solving the rate control constrained optimization problem defined in Equation 3. The high-level intuition of this reward mechanism is that the agent attempts to outperform its own historical performance on the constrained objective over the course of the training. We adapt MuZero to perform self-competition and refer to this agent as *MuZero-Rate-Controller* (MuZero-RC). Note that this mechanism differs from MuZero's self-play as self-play runs two-player games with the latest version of the agent, while self-competition agent is competing against its own historical performance in a single player setting.

To track the historical performance of the agent, we maintain exponential moving averages (EMA) of PSNR and overshoot (bitrate - target β) for each unique [video, target bitrate β] pair over the course of the training in a buffer. When the agent newly encodes a video with a target bitrate β, the encoder computes the PSNR and overshoot of the episode (P_{episode} and O_{episode} respectively). The agent looks up the historical PSNR and overshoot (P_{EMA} and O_{EMA} respectively) for that [video, target bitrate β] pair from the buffer. We first compare O_{episode} with the O_{EMA}, and then compare P_{episode} with P_{EMA}. The return for the episode is set to ± 1 depending on whether the episode is better than the EMA-based historical performance according to equation 4. We also update the EMAs in the buffer for the [video, target bitrate β] pair with the P_{episode} and O_{episode}. Algorithm 1 provides a detailed pseudocode of this self-competition algorithm.

$$\text{Return} = \underbrace{\text{sgn}(P_{\text{episode}} - P_{\text{EMA}})}_{\text{Improve PSNR}} \underbrace{\mathbb{1}_{O_{\text{EMA}} \leq 0, O_{\text{episode}} \leq 0}}_{\text{if constraints are satisfied}} - \underbrace{\text{sgn}(\lfloor O_{\text{episode}} \rfloor_+ - \lfloor O_{\text{EMA}} \rfloor_+)}_{\text{otherwise, ensure constraints are satisfied}} \tag{4}$$

where sgn is the sign function, $\mathbb{1}$ is the indicator function, and $\lfloor x \rfloor_+ = \max(x, 0)$.

To understand why this solution is suitable for variable difficulty constrained RL, we can interpret the EMA-based historical performance as a baseline for the agent to beat. As the agent learns to beat this baseline, the baseline itself improves via EMA updates. This leads the agent to improve its performance further to beat the newer baseline, which creates a cycle of performance improvement. Since the performance is measured by comparing bitrate constraint satisfaction first, the agent learns to satisfy the constraint first, and then learns to improve the PSNR. Unlike Lagrangian relaxation methods, this mechanism doesn't contain any direct parameters which represent the difficulty of constraint satisfaction. This allows the agent to infer the difficulty of satisfying the constraints from the observations.

This formulation can be related to applying Lagrangian relaxation methods on an augmented CMDP. The state of the CMDP can be augmented with EMAs, and the constrained objective becomes

$$\begin{aligned}\max_{\pi(\mathrm{QP}_t|s_t, P_{\mathrm{EMA}}, O_{\mathrm{EMA}}, \beta_{\mathrm{Target}})} \quad & \mathrm{sgn}(P_{\mathrm{episode}} - P_{\mathrm{EMA}})\mathbb{1}_{O_{\mathrm{EMA}} \leq 0, O_{\mathrm{episode}} \leq 0} \\ \text{s.t.} \quad & \mathrm{sgn}(\lfloor O_{\mathrm{episode}} \rfloor_+ - \lfloor O_{\mathrm{EMA}} \rfloor_+) \leq 0\end{aligned}$$

Using a Lagrangian relaxation method with $\lambda = 1$ on this objective results in the same return as equation 4. Reward mechanisms with historical performance based self-competition have been proposed in the past [Laterre et al., 2018, Schmidt et al., 2019], although they haven't been applied to constrained RL to the best of our knowledge.

4.3 Agent Architecture

Our training setup closely follows Schrittwieser et al. [2020]. We train this agent in the canonical asynchronous distributed actor-learner setting with experience replay [Lin, 1992, Horgan et al., 2018]. We maintain a shared buffer across all actors to track agent's EMA-based historical performance for each [video, target bitrate] pair. Actor processes sample [video, target bitrate] pairs randomly from the training dataset, and generate QPs for encoding them using the latest network parameters and 200 simulations of MuZero's MCTS algorithm. The self-competition based reward for the episode is computed using equation 4 and the episode is added to the experience replay buffer. The learner process samples transitions from the experience replay buffer to train the networks, and sends the updated parameters to the actor processes at regular intervals.

Features: At every step in the encoding process, the agent receives an observation from the encoder containing first-pass statistics for the video, PSNR and bits for the frames in the video encoded so far, index and type of the next frame to be encoded, and the target bitrate for the encoding. Appendix A.1 lists the features and preprocessing methods in detail.

Network Architecture: Similar to MuZero, the agent has three subnetworks. The representation network maps the features to an embedding of the state using Transformer-XL [Dai et al., 2019] encoders and ResNet-V2 style blocks [He et al., 2016]. The dynamics network generates the embedding of the next state given the embedding of the previous state and an action using ResNet-V2 style blocks. Given an embedding, the prediction network computes a policy using a softmax over the QPs, the value of the state, and several auxiliary predictions of the metrics related to the encoding using independent layer-normalised [Ba et al., 2016] feedforward networks. Appendix A.1 lists these auxiliary predictions and describes the architecture of the subnetworks in detail.

Training: At every step in the training, the learner uniformly samples a batch of states, five subsequent actions, and the necessary labels from the experience replay buffer. The representation network generates the embedding of the state, and the dynamics network is unrolled five times to generate the subsequent embeddings. The prediction network outputs the predictions for policy, value, and auxiliary metrics for the current and the five subsequent states. The policy prediction $\hat{\pi}_t$ is trained to match the MCTS policy generated at acting time π_t using cross-entropy loss. The value prediction $\hat{v}_t$ is trained to match the self-competition reward v_t for the episode using IQN loss [Dabney et al., 2018a], and the auxiliary predictions $\hat{y}_t$ are trained to match the corresponding metrics y_t using a quantile regression loss [Dabney et al., 2018b]. We use equation 5 to combine the losses.

$$\mathrm{Loss} = \frac{1}{6}\sum_{t=0}^{5}\left[\underbrace{L_{\mathrm{CE}}(\pi_t, \hat{\pi}_t)}_{\text{Policy}} + \underbrace{0.5 \cdot L_{\mathrm{IQN}}(v_t, \hat{v}_t)}_{\text{Value}} + 0.1 \cdot \sum_{\text{Auxiliary}} L_{\mathrm{QR}}(y_t, \hat{y}_t)\right] + \underbrace{10^{-3} \cdot \|\theta\|_2^2}_{\text{L2-Reg}} \quad (5)$$

In our experiments, we found the auxiliary losses crucial for learning the dynamics of the encoding in order to perform the MCTS. The hyperparameters of the training are described in appendix A.2.

4.4 Augmented MuZero-RC

We also propose a small variant of the MuZero-RC agent. We augment the self-competition mechanism in equation 4 by replacing the PSNR term with 'PSNR $- 0.005 \times$ overshoot' in order to get the agent to reduce bitrate if it can't improve PSNR. We label this agent 'Augmented MuZero-RC' and report the results alongside MuZero-RC in section 5. The factor of 0.005 was selected as it is roughly the average slope of VP9's RD curve across the training dataset. In our experiments, we didn't find significant performance difference with small adjustments to this factor.

5 Experiments

In this section, we first describe the experiment setup. This includes an overview of the environment, the dataset of videos used for training and evaluation, and the evaluation metrics. We then describe the performance of MuZero-RC agents in this setup compared to the baselines.

5.1 Setup

Environment setup and baseline: We use the libvpx implementation of VP9 in our experiments. In particular, we implemented an RL environment based on the SimpleEncode API [WebM, 2019] in libvpx. This allows us to override the libvpx's QP decision with an external QP computed by our agent. We train and evaluate the learned rate control policy using the environment. In our experiments, we use libvpx's two-pass variable bitrate (VBR) mode [WebM, 2017] at encode speed 1 (i.e., `--cpu-used=1`), which is a commonly used setting for video-on-demand (VOD). We use libvpx's default rate control implementation as our baseline as it is canonically used in VP9.

Dataset: We limit the focus of our experiments to 480p videos with target bitrate in [256, 768] kbps range, although our methods can be generalised to other resolutions and bitrates. For training, we use 20,000 randomly selected 480p videos that have varying duration (3 to 7 seconds), varying aspect ratios, varying frame rates, and diverse content types. The target bitrate for training is uniformly sampled from {256, 384, 512, 640, 768} kbps. For evaluation, we use videos with resolution 480p and higher from the YouTube UGC dataset [Wang et al., 2019], designed for benchmarking video compression and quality assessment research available under CC-BY 3.0 license. We resize the videos to 480p, and chunk them into 5-second clips, resulting in 3062 clips.

Metrics: To evaluate the coding efficiency and overall bitrate reduction, we measure the Bjontegaard-delta rate (BD-rate) [Bjontegaard, 2001] using libvpx's default VBR rate control policy as the reference. Given the bitrate v.s. PSNR curves of two policies, BD-rate computes the average bitrate difference for the same PSNR across the overlapped PSNR range, and therefore, measures the average bitrate reduction for encoding videos at the same quality. See figure 1b for an example. We report the mean PSNR BD-rate difference compared to libvpx on the evaluation set, as well as BD-rate difference with respect to perceptual quality measures SSIM and VMAF. To evaluate the bitrate constraint satisfaction, we report the fraction of evaluation cases on which the policies violate the bitrate constraint. Since a small amount of constraint violation is acceptable in practice, we also report the fraction of cases on which the policies violate the constraint by >5% of the target bitrate. We also note that since PSNR and bitrate are competing objectives, the optimal behavior is to use the target bitrate budget fully in order to maximize the PSNR. To evaluate this, we report the fraction of cases on which the policies achieve bitrate within 5% of the target. We report the histograms of BD-rate differences and bitrates in appendices A.6 and A.7.

Evaluation: We use the final network checkpoint generated by the training process for evaluation. Practically, the rate control step should not take a large amount of time. So, instead of performing MuZero-style MCTS, we select the QPs with the highest probability assigned by the policy output of the prediction network. We iterate over the evaluation dataset and encode each video with libvpx rate control and both the MuZero-RC agents at nine target bitrate values uniformly spaced between [256, 768] kbps. We evaluate the constraint satisfaction behavior of the agent for all nine target bitrates, and use the RD-curve generated by these encodings for each video to compute the BD-rate difference.

5.2 Results

PSNR BD-Rate improvement: First, we evaluate the PSNR BD-rate of MuZero-RC agents compared to libvpx. Table 1 shows the difference of PSNR BD-rate achieved by MuZero-RC over the evaluation set videos compared to libvpx's rate control. It shows that MuZero-RC can achieve better PSNR v.s. bitrate tradeoff with 4.72% average bitrate savings for encoding videos at same quality as libvpx, and Augmented MuZero-RC increases it further to 6.28%. The agent reduces the BD-rate with respect to perceptual quality measures SSIM and VMAF as well. We present the histogram of BD-rate differences over the evaluation video set in appendix A.6.

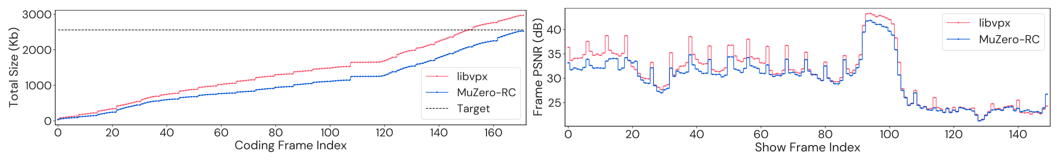
Table 1: BD-rate difference of MuZero-RC agents compared to libvpx with respect to various quality measures. We report ±1 standard error interval computed over 5 random seeds. As seen in the table, both MuZero-RC and Augmented MuZero-RC lead to BD-rate reductions compared to the libvpx baseline with respect to PSNR, SSIM and VMAF metrics.

Agent	Mean PSNR BD-rate difference	Mean SSIM BD-rate difference	Mean VMAF BD-rate difference
MuZero-RC	-4.72% ± 0.32%	-3.68% ± 0.33%	-0.53% ± 0.21%
Augmented MuZero-RC	**-6.28%** ± 0.19%	**-5.11%** ± 0.24%	**-1.88%** ± 0.12%

Constraint satisfaction: Then, we evaluate the bitrate constraint satisfaction behaviour. Table 2 lists the constraint satisfaction metrics achieved by libvpx baseline policy and the MuZero-RC agents. The MuZero-RC agent violates the constraints on substantially fewer videos compared to libvpx. Even when the constraints are violated, large magnitude violations by MuZero-RC are less frequent compared to libvpx. Also, MuZero-RC agent encodes videos with bitrates within 5% margin around the target bitrate more frequently than libvpx. This indicates that MuZero-RC not only overshoots less, but it also tends to fully use the target bitrate budget more frequently compared to libvpx. We also see that Augmented MuZero-RC satisfies constraints just as well as MuZero-RC, and its bitrate accuracy is comparable to libvpx. We present the histogram of overshoots over the evaluation video set for different target bitrates in appendix A.7. Figure 2 demonstrates a typical behavior of MuZero-RC regarding bitrate control. As shown in Figure 2a, the libvpx baseline overshoots the target bitrate by about 5%, while MuZero-RC is accurate in allocating the target bitrate (marked by black line). MuZero-RC avoids overshooting by using fewer bits (and thus sacrificing the PSNR) in early frames of the video as shown in Figure 2b. This is a good decision because the PSNR of the first part of the video is relatively high. In fact, if a policy sacrifices the PSNR in the last 40 frames, it would hurt the overall PSNR. This example highlights agent's ability to reason about frame complexity and bitrate-PSNR tradeoff, and plan accordingly.

Table 2: Bitrate constraint satisfaction behavior over the evaluation set. We report ±1 standard error interval computed over 5 different random seeds for MuZero-RC agents. As libvpx rate control is a deterministic process, we do not report the standard error intervals for it. As seen in the table, both MuZero-RC and Augmented MuZero-RC overshoot the target bitrate for substantially fewer test cases compared to libvpx. MuZero-RC agent achieves higher bitrate accuracy compared to libvpx while Augmented MuZero-RC achieves a comparable accuracy.

	Fraction of videos with		
Agent	overshoot > 0	overshoot > 5% of target	bitrate within 5% of target
libvpx	64.00%	6.13%	71.34%
MuZero-RC	20.34% ± 2.16%	**2.04%** ± 0.20%	**84.14%** ± 0.68%
Augmented MuZero-RC	**16.10%** ± 2.06%	2.25% ± 0.15%	70.12% ± 1.15%



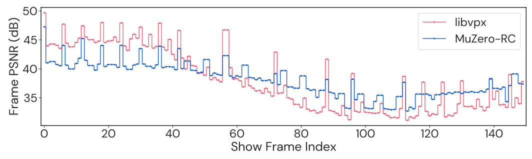
(a) Cumulative frame bits trajectories.

(b) Show frame PSNR trajectories.

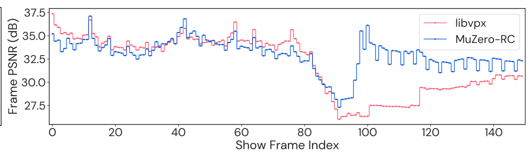
Figure 2: Encoding trajectories of video `LyricVideo_480P-1484`. The learned policy sacrifices PSNR in early frames in order to save bits to avoid bitrate overshooting in the end.

Planning performance: We investigate the encoding trajectories of videos and observe a few recurring patterns where MuZero-RC demonstrates better planning performance than libvpx. Figure

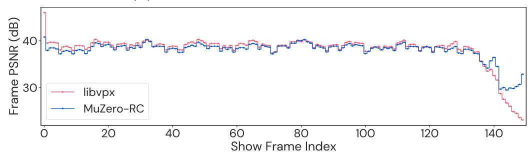
3 shows examples where MuZero-RC achieves better BD-rate. In Figure 3a, MuZero-RC sacrifices the PSNR of the first 60 frames in order to improve the PSNR of the rest of the frames. Because the latter part of the video is more complex (indicated by lower PSNR), improving the PSNR of the latter part of the video improves the overall PSNR. In Figure 3b, MuZero-RC boosts PSNR of a reference frame, leading to significant PSNR improvement for all the frames following that frame. In Figure 3c, the video has a scene change towards the very end. Because libvpx runs out of bitrate budget, it decides to encode the last few frames with low quality. In contrast, MuZero-RC correctly anticipates the scene change, and saves some bitrate budget from the first 140 frames so that it can obtain a reasonable PSNR for the last few frames. Figure 3d shows another pattern where MuZero-RC has overall less frame PSNR fluctuation. In particular, it avoids having extremely low PSNR on some frames, which in turn improves the overall PSNR.



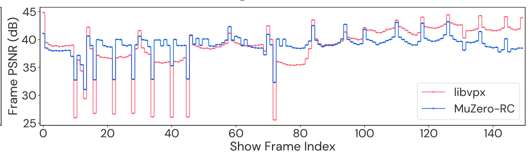
(a) `Animation_480P-073c`.



(b) `Gaming_1080P-26dc`.



(c) `CoverSong_720P-05d6`.



(d) `LiveMusic_480P-58fb`.

Figure 3: Show frame PSNR trajectories of 4 video clips.

6 Conclusions

In this paper, we have demonstrated that the MuZero reinforcement learning algorithm can be used for rate control in VP9. Our formulation of the self-competition based reward mechanism allows the agent to tackle the complex constrained optimization task and achieve better quality-bitrate tradeoff and better bitrate constraint satisfaction than libvpx's VBR rate control algorithm. The final agent results in 6.28% average reduction in bitrate (measured as PSNR BD-rate) on videos from the evaluation set, and can be readily deployed in libvpx via the SimpleEncode API.

Limitations: The self-competition based reward mechanism requires that every unique [video, target bitrate] pair be encoded a few times so that the historical performance converges and provides a reasonable baseline for reward computation. Because of this, the amount of data the actors need to generate increases linearly with the number of videos in the training dataset and the number of target bitrate samples. For very large training datasets, this method might not scale well. However, in future work, it may be possible to learn these baseline values based on observations using a neural network which can generalise to unseen videos in a large dataset.

Future Work: Our proposed methods are agnostic to the specifics of VP9/libvpx, and they can potentially be generalized not only to other coding formats and implementations, but also to other components within video encoders such as block partitioning and reference frame selection. Our method also opens the possibility of allowing codec developers and users to develop new rate control modes. For example, we can replace PSNR with other video quality metrics such as VMAF. We can also modify the reward to minimize bitrate given a minimum PSNR constraint – which is similar to the constrained quality (CQ) mode in libvpx, but reinforcement learning is likely to learn a policy that has more precise control of the PSNR.

Acknowledgments

We would like to thank Balu Adsumilli, Ross Wolf, Yaowu Xu, James Bankoski, Nishant Patil, Marisabel Hechtman, Dan A. Calian, Luis C. Cobo, Chris Gamble, and David Silver for the insightful discussions about this work and for their support throughout the project.

References

J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. *CoRR*, abs/1705.10528, 2017.

E. Altman. *Constrained Markov decision processes*. Stochastic modeling. Chapman & Hall/CRC, 1999. ISBN 9780849303821. URL `http://books.google.com/books?id=3X9S1NM2iOgC`.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

G. Bjontegaard. Calculation of average PSNR differences between RD-curves. *VCEG-M33*, 2001.

S. Bohez, A. Abdolmaleki, M. Neunert, J. Buchli, N. Heess, and R. Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

D. A. Calian, D. J. Mankowitz, T. Zahavy, Z. Xu, J. Oh, N. Levine, and T. Mann. Balancing constraints and rewards with meta-gradient D4PG. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=TQt98Ya7UMP`.

L.-C. Chen, J.-H. Hu, and W.-H. Peng. Reinforcement learning for hevc/h.265 frame-level bit allocation. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5, 2018. doi: 10.1109/ICDSP.2018.8631541.

Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning, 2018.

Cisco. Cisco annual internet report (2018–2023) white paper, 2020. URL `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

W. Dabney, G. Ostrovski, D. Silver, and R. Munos. Implicit quantile networks for distributional reinforcement learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR, 10–15 Jul 2018a. URL `http://proceedings.mlr.press/v80/dabney18a.html`.

W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018b.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Y. Efroni, S. Mannor, and M. Pirotta. Exploration-exploitation in constrained mdps, 2020.

W. Gao, S. Kwong, Y. Zhou, and H. Yuan. Ssim-based game theory approach for rate-distortion optimized intra frame ctu-level bit allocation. *IEEE Transactions on Multimedia*, 18(6):988–999, 2016.

Google, 2018. Cloud TPU. `https://cloud.google.com/tpu/`, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

Z. He and S. K. Mitra. Optimum bit allocation and accurate rate control for video coding via p-domain source modeling. *IEEE Trans. Cir. and Sys. for Video Technol.*, 12(10):840–849, Oct. 2002. ISSN 1051-8215. doi: 10.1109/TCSVT.2002.804883. URL `https://doi.org/10.1109/TCSVT.2002.804883`.

T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: Composable gradient transformation and optimisation, in JAX!, 2020. URL `http://github.com/deepmind/optax`.

Y.-H. Ho, G.-L. Jin, Y. Liang, W.-H. Peng, and X. Li. A dual-critic reinforcement learning framework for frame-level bit allocation in hevc/h. 265. In *2021 Data Compression Conference (DCC)*, pages 13–22. IEEE, 2021.

D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

J. Hu, W. Peng, and C. Chung. Reinforcement learning for hevc/h.265 intra-frame rate control. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

Jiangtao Wen, M. Luttrell, and J. Villasenor. Trellis-based r-d optimal quantization in h.263+. *IEEE Transactions on Image Processing*, 9(8):1431–1434, 2000.

D. Kwon, M. Shen, and C. . J. Kuo. Rate control for h.264 video with enhanced rate and distortion models. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(5):517–529, 2007.

S. Kwong, M. Zhou, W. Xuekai, W. Jia, and B. Fang. Rate control method based on deep reinforcement learning for dynamic video sequences in hevc. *IEEE Transactions on Multimedia*, 2020.

A. Laterre, Y. Fu, M. K. Jabri, A.-S. Cohen, D. Kas, K. Hajjar, T. S. Dahl, A. Kerkeni, and K. Beguir. Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. *arXiv preprint arXiv:1807.01672*, 2018.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

B. Li, H. Li, L. Li, and J. Zhang. λ domain rate control algorithm for high efficiency video coding. *IEEE transactions on Image Processing*, 23(9):3841–3854, 2014.

Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara. Toward a practical perceptual video quality metric, 2016. URL `https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652`.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

S. Ma, Wen Gao, and Yan Lu. Rate-distortion analysis for h.264/avc video coding and its application to rate control. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(12):1533–1544, 2005.

H. Mao, C. Gu, M. Wang, A. Chen, N. Lazic, N. Levine, D. Pang, R. Claus, M. Hechtman, C.-H. Chiang, C. Chen, and J. Han. Neural rate control for video encoding using imitation learning, 2020.

Minqiang Jiang and Nam Ling. On enhancing h.264/avc video rate control by psnr-based frame complexity estimation. *IEEE Transactions on Consumer Electronics*, 51(1):281–286, 2005.

A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje. The latest open-source video codec VP9 - an overview and preliminary results. In *2013 Picture Coding Symposium (PCS)*, pages 390–393, 2013. doi: 10.1109/PCS.2013.6737765.

S. Paternain, L. Chamon, M. Calvo-Fullana, and A. Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, pages 7555–7565, 2019.

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

S. Sanz-Rodriguez and F. Diaz-de-Maria. Rbf-based qp estimation model for vbr control in h.264/svc. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(9):1263–1277, 2011.

D. Schmidt, N. Moran, J. S. Rosenfeld, J. Rosenthal, and J. Yedidia. Self-play learning without a reward metric. *arXiv preprint arXiv:1912.07557*, 2019.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839):604–609, 2020.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018. URL `https://mitpress.mit.edu/books/reinforcement-learning-second-edition`.

C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

C. Tessler, D. J. Mankowitz, and S. Mannor. Reward constrained policy optimization. *CoRR*, abs/1805.11074, 2018. URL `http://arxiv.org/abs/1805.11074`.

Tihao Chiang and Ya-Qin Zhang. A new rate control scheme using quadratic rate distortion model. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 2, pages 73–76 vol.2, 1996.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Y. Wang, S. Inguva, and B. Adsumilli. Youtube ugc dataset for video compression research. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, 2019.

Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

WebM, 2010a. URL `https://chromium.googlesource.com/webm/libvpx/+/master/vp9/encoder/vp9_firstpass.h`.

WebM, 2010b. URL `https://chromium.googlesource.com/webm/libvpx`.

WebM, 2017. URL `https://developers.google.com/media/vp9/bitrate-modes/`.

WebM, 2019. URL `https://chromium.googlesource.com/webm/libvpx/+/master/vp9/simple_encode.h`.

R. Zhang, T. Yu, Y. Shen, H. Jin, C. Chen, and L. Carin. Reward constrained interactive recommendation with natural language feedback, 2020.

M. Zhou, X. Wei, S. Kwong, W. Jia, and B. Fang. Rate control method based on deep reinforcement learning for dynamic video sequences in hevc. *IEEE Transactions on Multimedia*, pages 1–1, 2020.

A Appendix

A.1 Network Architecture

As required by the MuZero algorithm, the MuZero-RC agent has three subnetworks.

Representation Network: This network takes the features provided by the environment as the input and produces an embedding of the current state of the environment as the output. For any state, the environment generates following observations.

1. A sequence of first-pass statistics for all the show frames in the video.
2. A sequence of PSNR, number of used bits, and applied QPs for all the previously encoded frames in the video so far, along with indices of those frames.
3. The index and the type of the frame to be encoded next. The type can be one of five frame types from the SimpleEncode API.
4. The duration of the video.
5. Target bitrate for the encoding.

We use the same first pass statistics and features normalization methods used by Mao et al. [2020]. Additionally, we generate the fraction of the target bitrate used so far in the encoding using the bits used by previously encoded frames and video duration. We use this fraction as an additional scalar feature.

The representation network aligns the first two sequential features along the indices of the frames and concatenates the aligned sequences along the feature dimension. This concatenated sequence is then processed using a series of 4 Transformer-XL encoder blocks [Dai et al., 2019]. From this sequence, the entry at index of the frame to be encoded next is extracted. This entry is concatenated with the remaining scalar features and processed using two feedforward layers with intermediate layer normalization [Ba et al., 2016]. The network processes the output of these layers with a series of 4 layer normalized ResNet-V2 blocks [He et al., 2016]. The output of these blocks is the embedding of the state. We use an embedding of 512 units in all our experiments. All the layers use ReLU as the activation function. Figure 4 shows a diagram of this network.

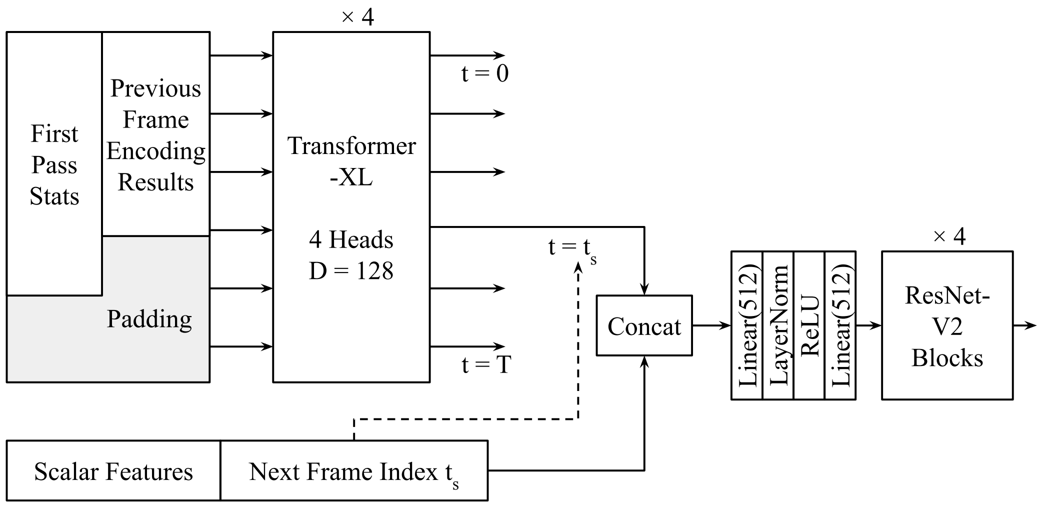


Figure 4: MuZero-RC Representation Network

Dynamics Network: This network takes an embedding of the state and the QP to be applied in that state as the input. It produces an embedding of the next state reached after applying the QP as output. This network processes the QP using two feedforward layers with intermediate layer normalization to output a vector with the same dimension as the embedding of the previous state. It performs elementwise addition of this vector and the embedding of the previous state, and processes the result with a series of 4 layer normalized ResNet-V2 blocks. The output of these blocks is the embedding of the next state reached after applying the QP. All the layers use ReLU as the activation function. Figure 5 shows a diagram of this network.

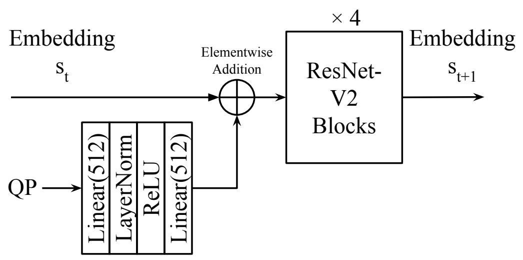


Figure 5: MuZero-RC Dynamics Network

Prediction Network: This network takes the embedding of a state and produces the policy, value, and several auxiliary predictions as the output. For the policy prediction, the network processes the state embedding with two feedforward layers with 256 hidden units and layer normalization followed by a linear layer of 256 units representing the logits for each QP value. A softmax function is applied to these logits to produce the policy. For the value prediction, the network processes the state embedding with two feedforward layers with 256 hidden units and layer normalization followed by a linear layer of 64 units. The output of this layer is used as an embedding for the IQN layer which produces samples of the value prediction. We apply the tanh function to these samples to limit them in range (-1, 1) as the value in the self-competition based reward mechanism is limited to [-1, 1]. At training time, we draw 8 samples from the IQN layer to match the self-competition reward. At inference time, we use the expected value instead of sampling.

For each of the auxiliary predictions, the network processes the state embedding with two feedforward layers with 256 hidden units and layer normalization followed by a linear layer of 64 units. The output of this layer represents the uniformly spaced quantiles of the corresponding auxiliary prediction. In our experiments, we predict the following metrics as auxiliary predictions.

1. The PSNR of the last encoded frame (0 when no frames are encoded).
2. The log of the number of bits used by the last encoded frame (0 when no frames are encoded).
3. The expected PSNR of the video being encoded.
4. The expected bitrate of the video being encoded.

These auxiliary predictions help the agent understand the dynamics of the video encoding process, which we found to be crucial in our experiments. All the layers use ReLU as the activation function unless specified otherwise. Figure 6 shows a diagram of this network.

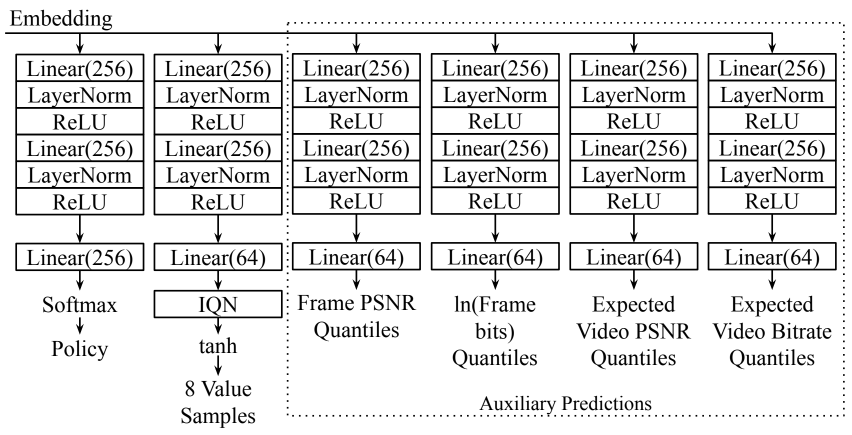


Figure 6: MuZero-RC Prediction Network

All experiments in this paper are implemented using the JAX framework [Bradbury et al., 2018]. We use the Haiku library to implement the subnetworks [Hennigan et al., 2020].

A.2 Training

We train all the subnetworks jointly similar to Schrittwieser et al. [2020] in order to minimize the loss in equation 5. We use the SGD algorithm with momentum factor of 0.9 for optimization. The learning rate for the optimization reduces over the course of training as follows

$$\text{lr}(t) = \text{lr}_{\text{init}} \cdot \text{decay}^{\frac{t}{\text{interval}}}$$

where t is the training step, $\text{lr}_{\text{init}} = 0.05$, decay is 0.1 and interval is 300,000. We use the Optax library [Hessel et al., 2020] for optimization. We train the agent for 1 million steps. 400 million frames are generated by the actor processes over the course of training. For experience replay, we keep a buffer of 50,000 latest episodes generated by the actor processes, and draw samples with batch size 512 from the replay buffer.

A.3 Computational Resources

All experiments in this paper are run using Google Cloud TPUs [Google, 2018]. The learner processes use two 3rd generation TPUs, and the actor processes use four 2nd generation TPUs for 15 hours. Additionally, we use 3000 CPU cores from a shared heterogeneous compute cluster for running video encoding with libvpx for actors.

A.4 Lagrangian-relaxation Method for Rate Control

To evaluate the hypothesis that Lagrangian-relaxation methods for constrained RL are not suitable for rate control, we train a MuZero agent with Lagrangian-relaxation based value function similar to Tessler et al. [2018]. We maintain all other hyperparameters and architecture same as MuZero-RC apart from the tanh activation on value network, which is removed in this agent as the value is no longer in [-1, 1]. We observe that this agent was difficult to tune, and we had to run multiple hyperparameter sweeps to get a stable instance. We evaluated this instance using the protocol described in section 5 and report the metrics in tables 3 and 4 respectively. We notice that the agent exhibits poor performance compared to libvpx in terms of the PSNR BD-rate reduction. We also note that while the agent satisfies the constraint for a reasonable number of test videos, it tends to use very little of the bitrate target budget as seen in the last column of table 4. Also, the number of videos on which agent overshoots by >5% of the target is significantly higher compared to the other policies.

Table 3: BD-rate difference of Lagragnian-relaxation method compared to libvpx. Columns in this table are same as those in table 1.

Agent	Mean PSNR BD-rate difference	Mean SSIM BD-rate difference	Mean VMAF BD-rate difference
Lagrangian	256.37%	41.44%	195.12%
MuZero-RC	-4.72% ± 0.32%	-3.68% ± 0.33%	-0.53% ± 0.21%

Table 4: Bitrate constraint satisfaction behavior of Lagrangian-method over the evaluation set. Columns in this table are same as those in table 2.

	Fraction of videos with		
Agent	overshoot > 0	overshoot > 5% of target	bitrate within 5% of target
libvpx	64.00%	6.13%	71.34%
Lagrangian	25.83%	25.49%	0.68%
MuZero-RC	20.34% ± 2.16%	2.04% ± 0.20%	84.14% ± 0.68%

A.5 Pseudocode for Self-competition Reward

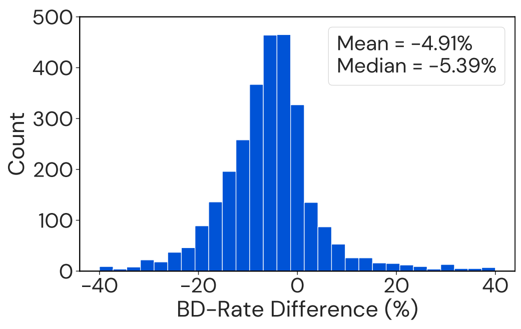
Algorithm 1 illustrates the pseudocode for the self-competition reward mechanism. In our experiments, we set the parameters $\text{PSNR}_0 = 30$ dB and $\alpha = 0.9$.

Algorithm 1 Self-competition reward mechanism for MuZero-RC

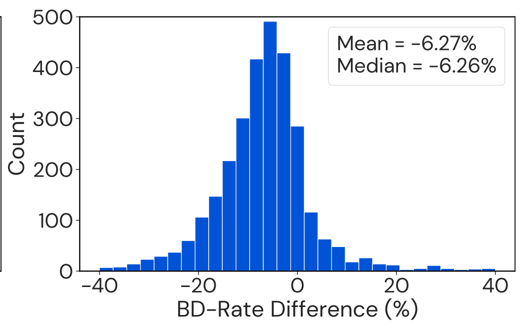
```
1: Dataset $\mathcal{D}$ of videos with assigned target bitrates
2: Buffer $\mathcal{B} \leftarrow [[\text{Video}_i, \beta_i] \rightarrow (\text{PSNR}_0, 0)\ \forall$ videos]
3: $\alpha :$ Buffer update factor
4: repeat                                                          ▷ Actor Process
5:     video$_i$, target_bitrate $\beta_i \sim \mathcal{D}$          ▷ Sample video and target bitrate from dataset
6:     PSNR$_\text{Episode}$, Bitrate$_\text{Episode}$ ← encode_video(video$_i|\beta_i$, agent policy)
7:                              ▷ Encode video using agent policy and compute PSNR and bitrate
8:
9:     UPDATE_BUFFER($\mathcal{B}$, video$_i$, PSNR$_\text{Episode}$, overshoot$_\text{Episode}$, $\beta_i$, $\alpha$)
10:    Return ← SELF_COMPETITION_REWARD($\mathcal{B}$, video$_i$, PSNR$_\text{Episode}$, overshoot$_\text{Episode}$, $\beta_i$)
11: until end
12:
13: procedure UPDATE_BUFFER($\mathcal{B}$, video$_i$, PSNR$_\text{Episode}$, overshoot$_\text{Episode}$, $\beta_i$, $\alpha$)
14:     $\mathcal{B}\,[\text{video}_i, \beta_i]\,[0] \leftarrow (1-\alpha) \cdot \mathcal{B}\,[\text{video}_i, \beta_i]\,[0] + \alpha \cdot \text{PSNR}_\text{Episode}$
15:     $\mathcal{B}\,[\text{video}_i, \beta_i]\,[1] \leftarrow (1-\alpha) \cdot \mathcal{B}\,[\text{video}_i, \beta_i]\,[1] + \alpha \cdot \text{overshoot}_\text{Episode}$
16: end procedure
17:
18: procedure SELF_COMPETITION_REWARD($\mathcal{B}$, video$_i$, PSNR$_\text{Episode}$, overshoot$_\text{Episode}$, $\beta_i$)
19:                                                              ▷ Equivalent to equation 4
20:     PSNR$_\text{EMA}$, overshoot$_\text{EMA}$ ← $\mathcal{B}\,[\text{video}_i, \beta_i]$
21:     if overshoot$_\text{Episode} > 0$ or overshoot$_\text{EMA} > 0$ then
22:         if overshoot$_\text{Episode} \leq$ overshoot$_\text{EMA}$ then
23:             return $1$
24:         else
25:             return $-1$
26:         end if
27:     else
28:         if PSNR$_\text{Episode} \geq$ PSNR$_\text{EMA}$ then
29:             return $1$
30:         else
31:             return $-1$
32:         end if
33:     end if
34: end procedure
```

A.6 Histograms of BD-Rate differences compared to libvpx

This section presents the histograms of BD-rate difference of the behaviors of MuZero-RC and Augmented MuZero-RC on 3062 videos from evaluation set compared to libvpx. See figure 1b for a qualitative example of RD curves and BD-rate difference. The RD curves for all the policies are computed by encoding each video from the evaluation set with nine uniformly spaced target bitrates from the [256, 768] kbps. The figures 7, 8, and 9 show the histogram of BD-rate differences for PSNR, SSIM, and VMAF respectively for a single run of both agents compared to libvpx.

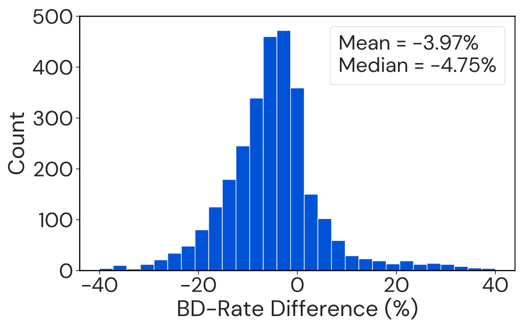


(a) MuZero-RC

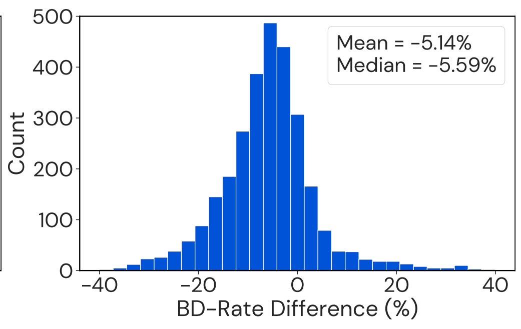


(b) Augmented MuZero-RC

Figure 7: Histogram of PSNR BD-Rate difference of agent compared to libvpx on evaluation set.

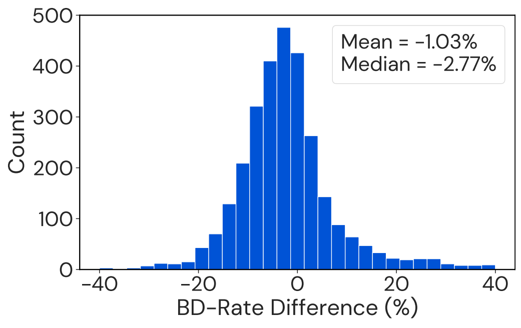


(a) MuZero-RC

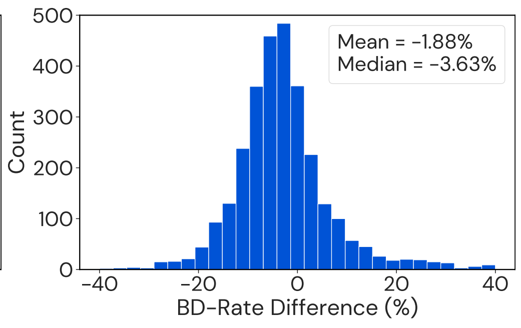


(b) Augmented MuZero-RC

Figure 8: Histogram of SSIM BD-Rate difference of agent compared to libvpx on evaluation set.



(a) MuZero-RC

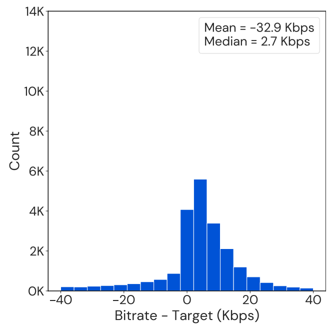


(b) Augmented MuZero-RC

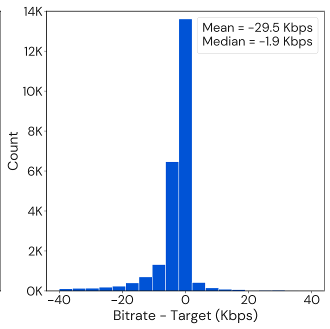
Figure 9: Histogram of VMAF BD-Rate difference of agent compared to libvpx on evaluation set.

A.7 Histograms of bitrate constraint satisfaction

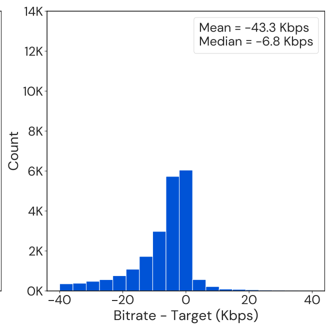
This section presents histograms of constraint satisfaction behaviors of libvpx, MuZero-RC, and Augmented MuZero-RC policies on 3062 videos from evaluation set. We encode the videos using each of these policies with nine uniformly spaced target bitrates from the [256, 768] kbps. We report the histograms of overshoot (bitrate - target) of the encodings done by each of the policies on aggregate as well as sliced by each of the target bitrate value. Figure 10 shows the histogram of the overshoots for all the policies with all the target bitrates together. Figures 11-19 show the histograms of the overshoots for all the policies with target bitrate of 256, 320, 384, 448, 512, 576, 640, 704, and 768 kbps respectively.



(a) libvpx

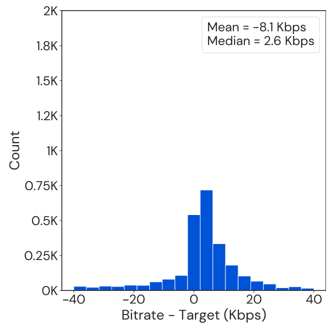


(b) MuZero-RC

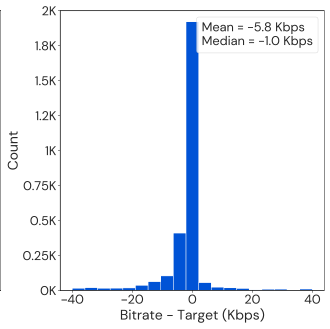


(c) Augmented MuZero-RC

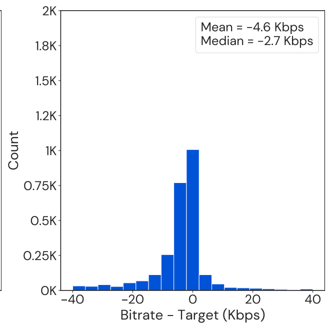
Figure 10: Histogram of overshoots of the agents on the evaluation set for all target bitrates.



(a) libvpx

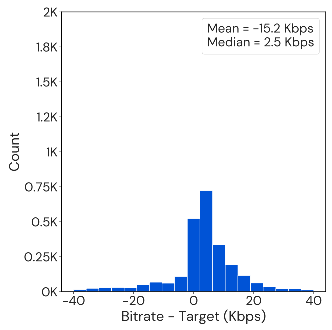


(b) MuZero-RC

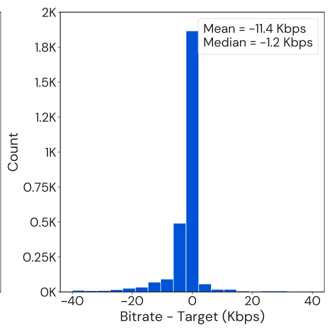


(c) Augmented MuZero-RC

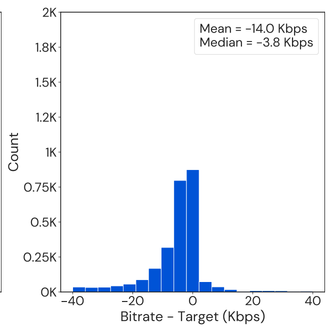
Figure 11: Histogram of overshoots of the agents on the evaluation set for 256 kbps target bitrate.



(a) libvpx

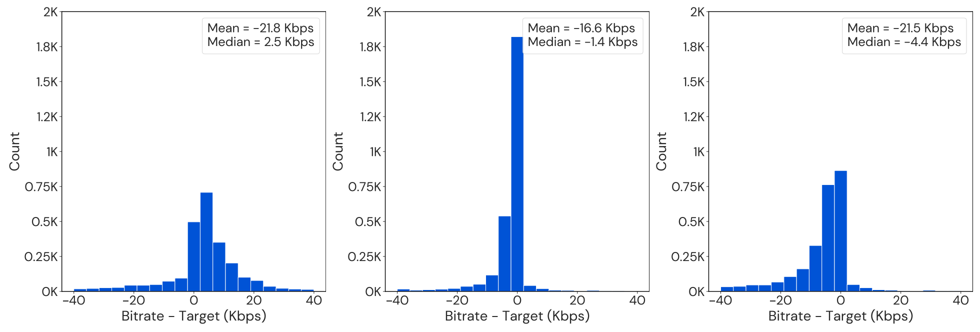


(b) MuZero-RC



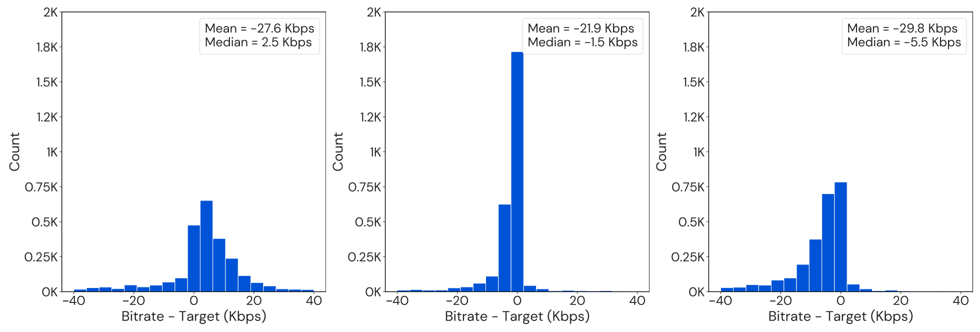
(c) Augmented MuZero-RC

Figure 12: Histogram of overshoots of the agents on the evaluation set for 320 kbps target bitrate.



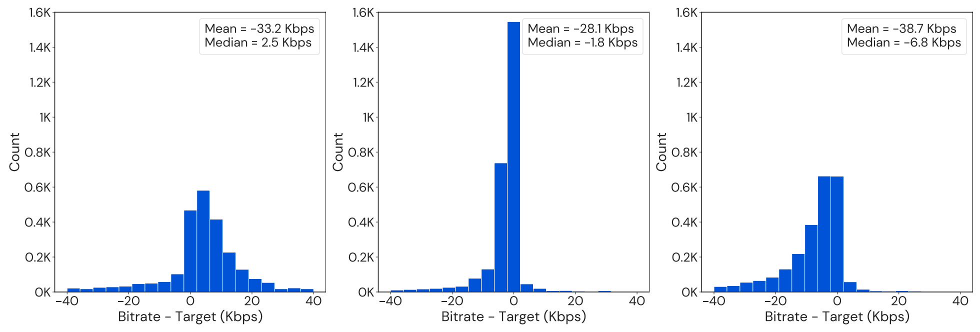
(a) libvpx (b) MuZero-RC (c) Augmented MuZero-RC

Figure 13: Histogram of overshoots of the agents on the evaluation set for 384 kbps target bitrate.



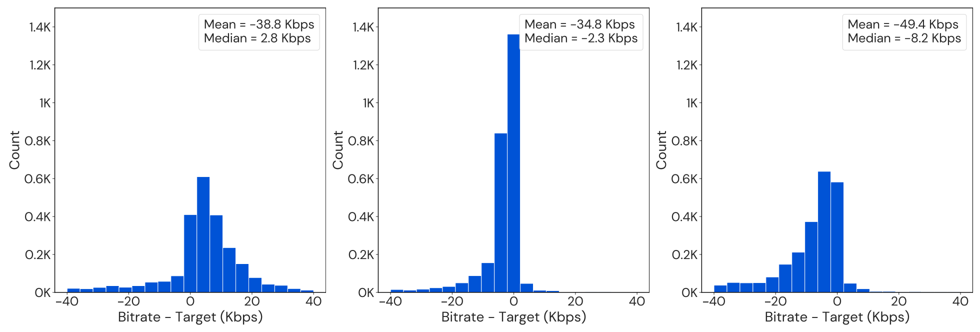
(a) libvpx (b) MuZero-RC (c) Augmented MuZero-RC

Figure 14: Histogram of overshoots of the agents on the evaluation set for 448 kbps target bitrate.



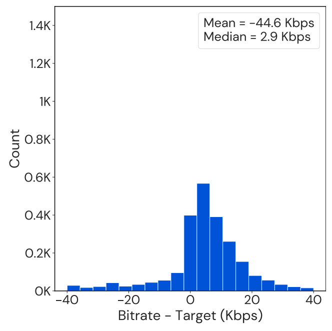
(a) libvpx (b) MuZero-RC (c) Augmented MuZero-RC

Figure 15: Histogram of overshoots of the agents on the evaluation set for 512 kbps target bitrate.

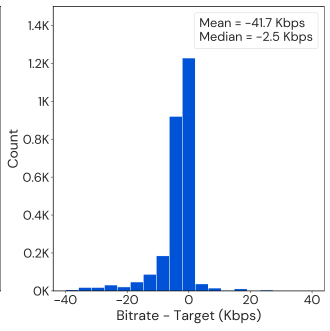


(a) libvpx (b) MuZero-RC (c) Augmented MuZero-RC

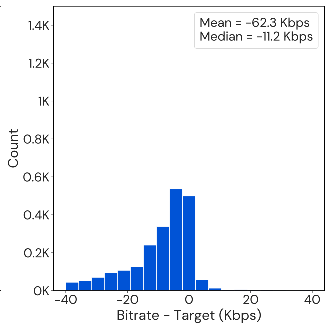
Figure 16: Histogram of overshoots of the agents on the evaluation set for 576 kbps target bitrate.



(a) libvpx

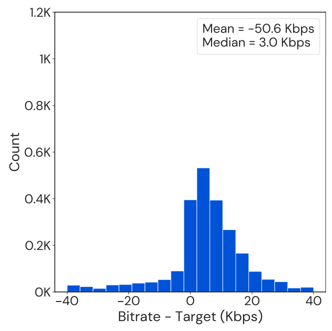


(b) MuZero-RC

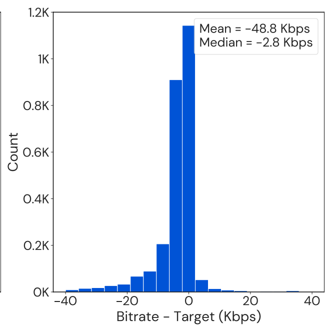


(c) Augmented MuZero-RC

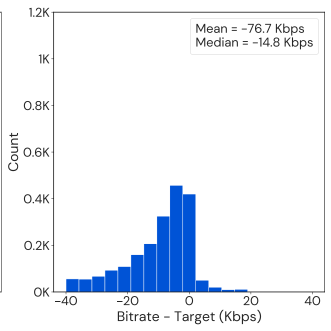
Figure 17: Histogram of overshoots of the agents on the evaluation set for 640 kbps target bitrate.



(a) libvpx

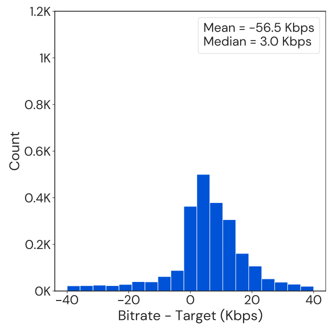


(b) MuZero-RC

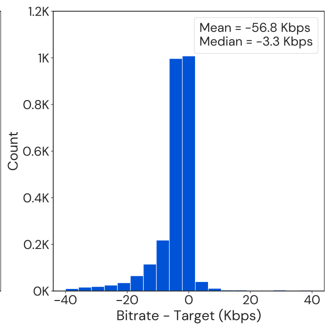


(c) Augmented MuZero-RC

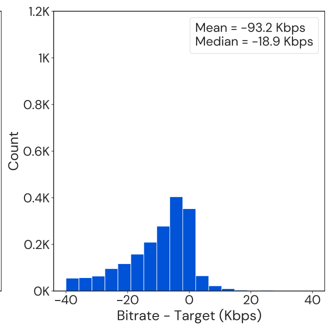
Figure 18: Histogram of overshoots of the agents on the evaluation set for 704 kbps target bitrate.



(a) libvpx



(b) MuZero-RC



(c) Augmented MuZero-RC

Figure 19: Histogram of overshoots of the agents on the evaluation set for 768 kbps target bitrate.