7. Appendix

7.1. Environment and agent details

7.1.1. Environment details

Global encoder statistics used as state information
Next frame x264 selected QP value
Next frame number
Current bitstream size
Current frame x264 selected QP value
Average QP
Percentages of I type Macro Blocks
Percentages of P type Macro Blocks
Percentages of skip-type Macro Blocks
x264 calculated PSNR
x264 calculated SSIM
Percentages of bits used for Motion Vectors
Percentages of bits used for DCT coefficient
Progress of encoding
bit-rate error
Next frame type
Next frame complexity

Table 4. Detailed components of global encoder statistic used in state information

Local (per-MB) encoder statistic used as state information
x264 energy values per Macro Block
x264 intra encoding cost per Macro Block
x264 propagating encoding cost per Macro Block
x264 inverse quantization scale factor per Macro Block

Table 5. Detailed components of per-MB encoder statistic used in state information

7.1.2. Agent architecture

To train the policy, we use the PPO algorithm [31], where the architecture of the policy is as follows: The per-block statistics are processed through a compact convolutional neural network (CNN) comprising three convolutional layers. These layers employ kernel sizes of $3x3$ or $4x4$ with a stride of 1. The resulting features are subsequently flattened and concatenated with the global statistics. A fully connected layer then derives a latent representation of dimension 64. This latent representation serves as input to three distinct fully connected networks: the value network (critic), the policy network (actor), and the reward prediction network described in the following subsection. A diagram of the full system is given in Figure 7. The agent's stochastic policy is modeled using a diagonal multivariate Gaussian distribution, where the agent learns the state-dependent mean vectors while maintaining independent standard deviation parameters for each dimension.

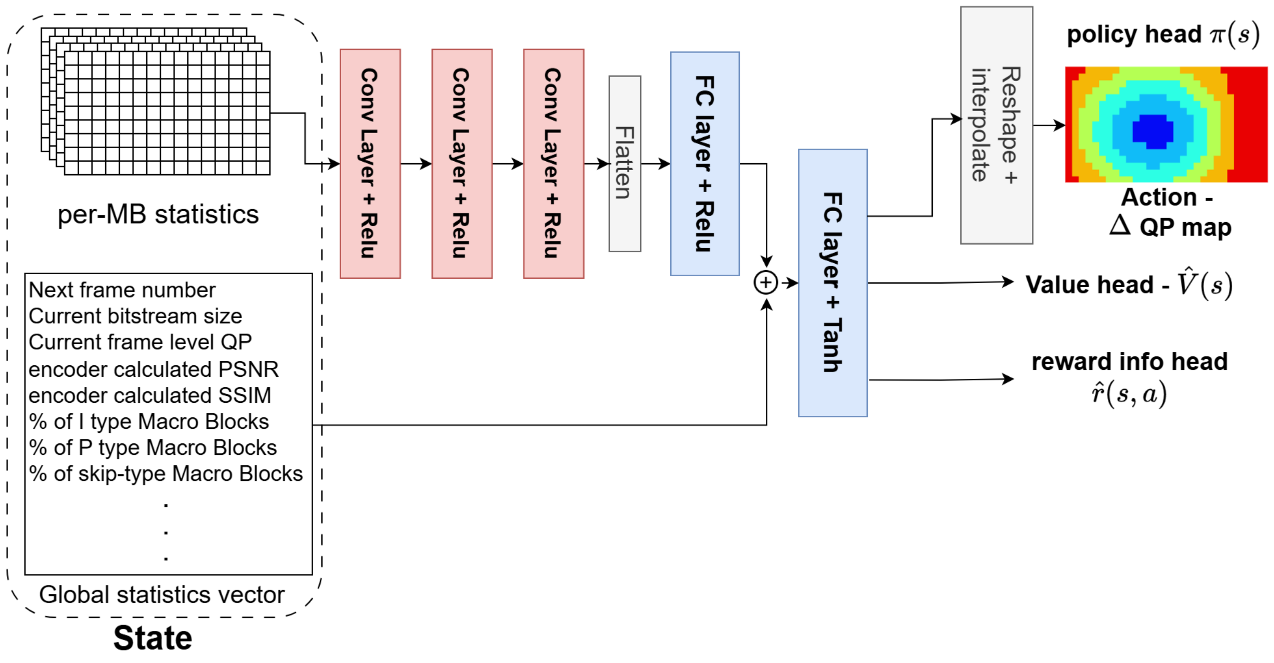

Figure 7. RL-RC-DoT agent architecture; Input is the statistics from the encoder, the output is the delta QP map

7.2. More details on RD-curves and BD-rate

The effectiveness of video compression is typically measured by comparing the compressed video's file size and visual quality to the original. Metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM; [39]) are often used to objectively assess quality. When comparing two encoders the compression efficiency is usually considered. To do so, a video is encoded in several desired bit-rates with each encoder to form a rate-distortion (RD) curve, where the y axis is the quality measure, e.g. PSNR. If one encoder's curve is shifted-left than the other, it means it requires less bits to reach the same quality, rendering it more efficient. If we integrate over the entire curve, and average the result over multiple videos, we obtain a quantity specifying how much bits saves one encoder than the other, a quantity referred to as Bjontegaard delta rate (BD-rate) [41].

With the increasing usages of videos for machine vision, many researchers have recognized the need for task-aware compression and proposed a suitable evaluation metric [16, 32]. The most straightforward metric which we also use in this paper is obtained by replacing the PSNR in the RD-curve (the y-axis) with a task-specific loss measure such as mIOU or detection precision.

7.3. Further results

Full results tables with bit-rate error: Here, we provide all of the tables used in the main text with the bit-rate error data.

ROI encoding experiment	**Saliency-weighted PSNR BD-rate**	**PSNR BD-rate**	**Bit-rate error** $[1e-3]$
RL-RC-DoT	-25.64 ± 0.99	-5.26 ± 0.36	-1.0 ± 0.43

Table 6. Results on RL-RC-DoT applied on the test-set for the saliency-weighted PSNR task.

	Precision (YOLO)	**Recall (YOLO)**	**PSNR**	**Precision (SSD)**
RL-RC-DoT	-24.7 ± 1.57	-19.75 ± 2.97	1.19 ± 0.46	-26.2 ± 1.48
	Recall (SSD)	**Segmentation IOU**	**Bit-rate error** $[1e-3]$	
RL-RC-DoT	-25.81 ± 2.03	-14.6 ± 1.81	0.13 ± 0.44	

Table 7. BD-rate Results on RL-RC-DoT applied on test set for the car detection task for various settings. Negative values mean that RL-RC-DoT improves over baseline.

Car detection	**Precision BD-rate**	**Recall BD-rate**	**PSNR BD-rate**	**Bit-rate error** $[1e-3]$
RL-RC-DoT	-24.7 ± 1.57	-19.75 ± 2.97	1.19 ± 0.46	0.13 ± 0.44
RL-RC-DoT w/o RI	-19.4 ± 1.38	-11.94 ± 1.7	1.92 ± 0.41	0.4 ± 0.47
RL-RC-DoT $\gamma = 0$	-9.78 ± 1.29	-10.28 ± 2.14	5.44 ± 0.6	2.4 ± 0.44
ROI encoding	**Saliency-weighted PSNR BD-rate**	**PSNR BD-rate**	**Bit-rate error** $[1e-3]$	
RL-RC-DoT	-25.64 ± 0.99	-5.26 ± 0.36	-1.0 ± 0.43	
RL-RC-DoT w/o RI	-23.46 ± 0.97	-4.54 ± 0.42	5.3 ± 0.48	
RL-RC-DoT $\gamma = 0$	-16.01 ± 0.77	2.11 ± 0.31	6.9 ± 0.43	

Table 8. Ablation study. (1) Full RL-RC-DoT (2) Omitting reward information (RI) from the training process and (3) Ignoring long term effects by using a myopic policy.

Here we give more qualitative results, Figures 9 and 9 gives more detection comparison between RL-RC-DoT and x264.

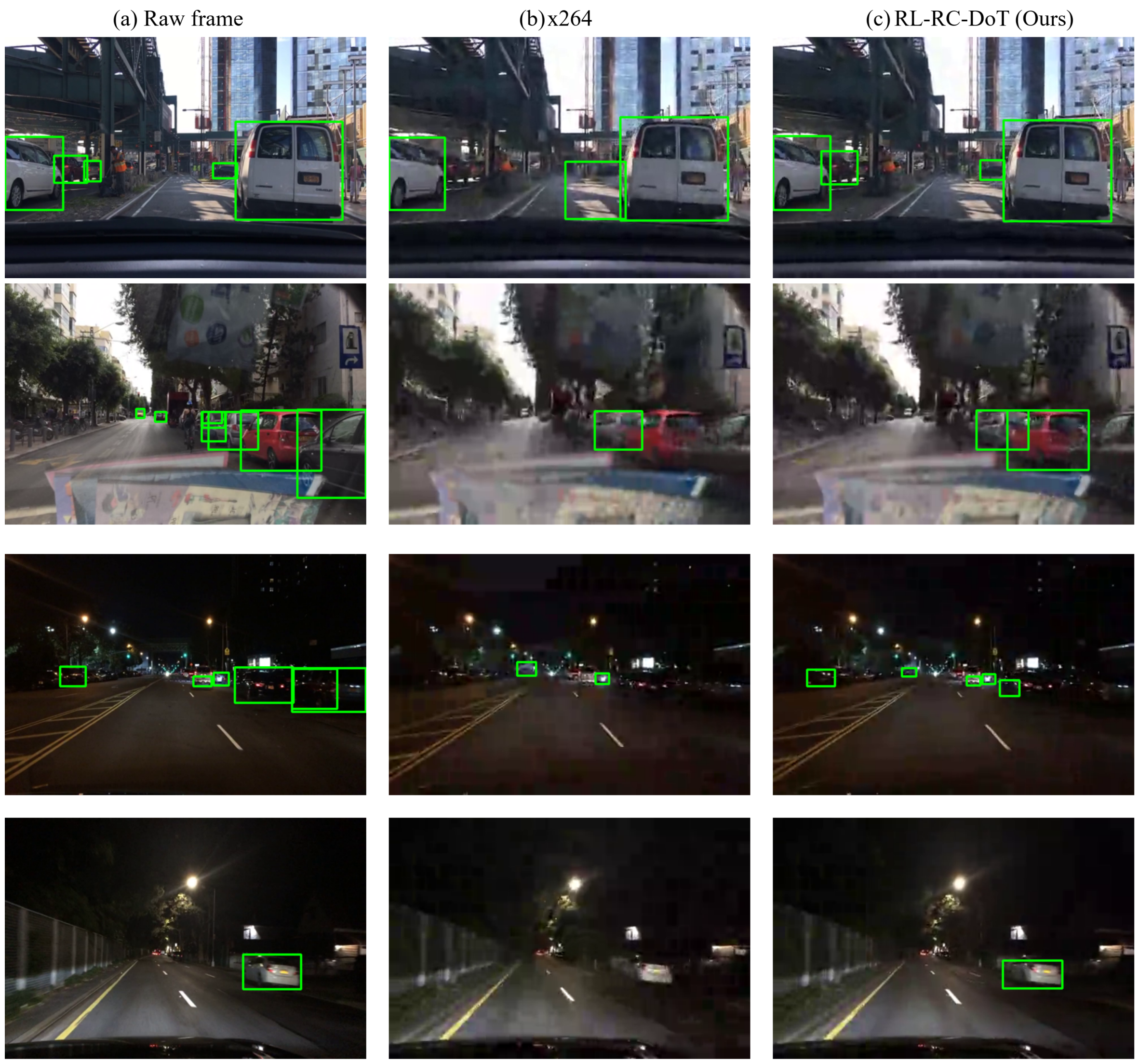

Figure 8. Car detection example result. (a) detection output on x264 reconstructed frame, (b) output on raw frame and (c) output on RL-RC-DoT reconstructed frame. Notice that both RL-RC-DoT and x264 used the same target bit-rate

Qualitative results on task-robustness: Figure 10, shows a qualitative example of task robustness. We compare the images in both types of rate-control, and the output of the downstream task. We can see the details corresponding to the downstream task are better reconstructed yielding a more relevant image.

7.3.1. Additional downstream tasks

We evaluated RL-RC-DoT with two more downstream tasks. First, in **video segmentation** using the DAVIS dataset [29]. RL-RC-DoT reduced BD-rate by 8% over x264. Quality was measured by IoU of a Mask R-CNN [11] segmentation. Second, in a **tracking** task, where the model was pre-trained for detection using the BDD dataset and tested in a different task of multi-object tracking (task shift). RL-RC-DoT reduced BD-rate by 3.2% over x264. Quality was measured by MOTA using ByteTracker [48].

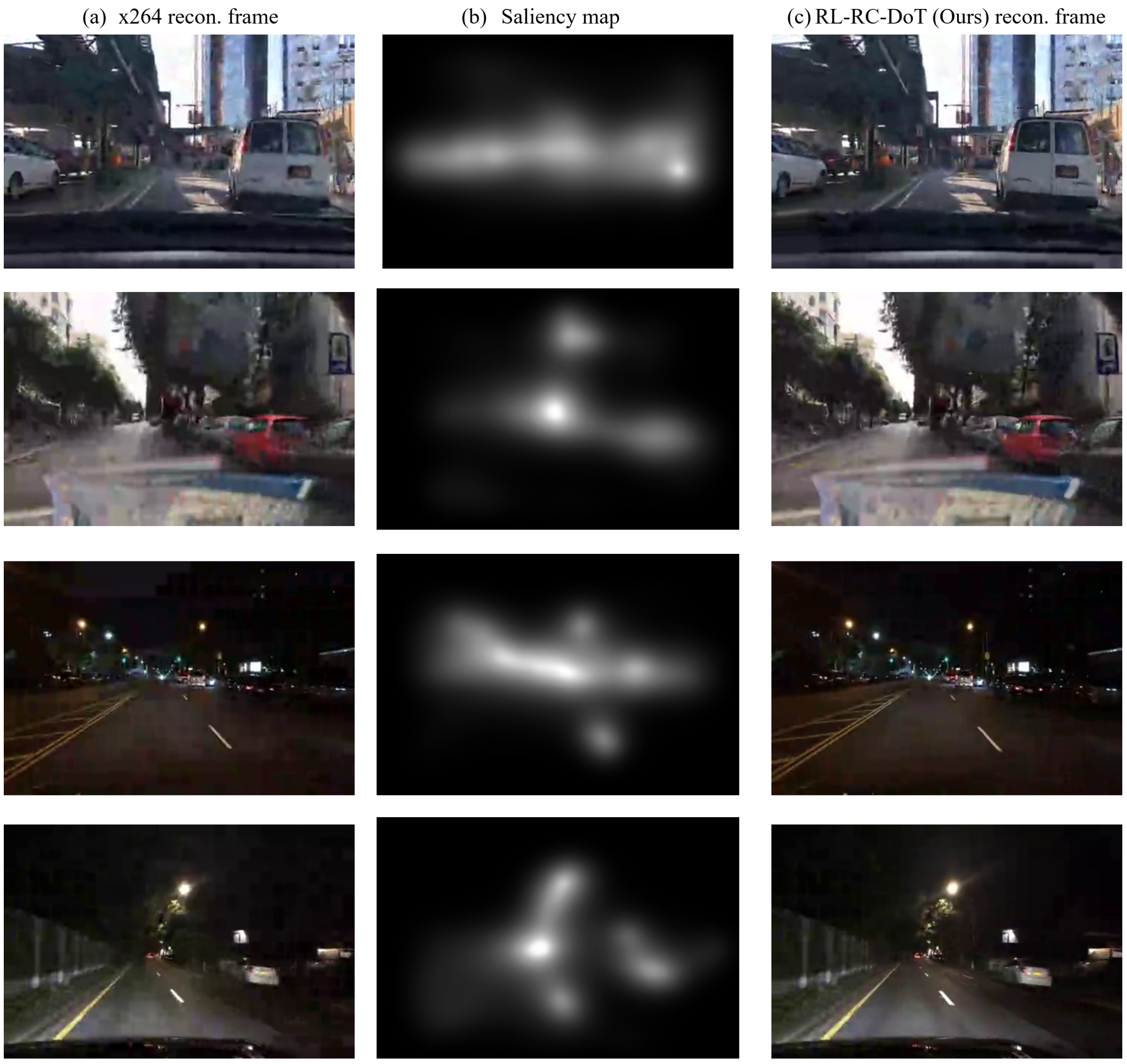

Figure 9. Saliency weighted PSNR results. (a) x264 reconstructed frame, (b) Saliency map of raw frame, extracted with [22] (c) RL-RC-DoT reconstructed frame. Notice that both RL-RC-DoT and x264 used the same target bit-rate

x264 preset	superfast	fast	veryfast	medium	slower	slow	veryslow
Precision BD-rate	-14.5	-22/8	-21.9	-24.7	-23.2	-25.2	-24.9

Table 9. Test BD-rate reduction of RL-RC-DoT for various x264 presets on car detection. Negative values mean that RL-RC-DoT improves over x264.

7.3.2. Different x264 configurations

As mentioned in section 4.4. We only trained our agent on the *medium* preset configuration of x264. To test RL-RC-DoT generalization palpabilities, we tested the agent on different preset configurations of x264. The BD-rate saving for car detecion precision are shown in table 9.

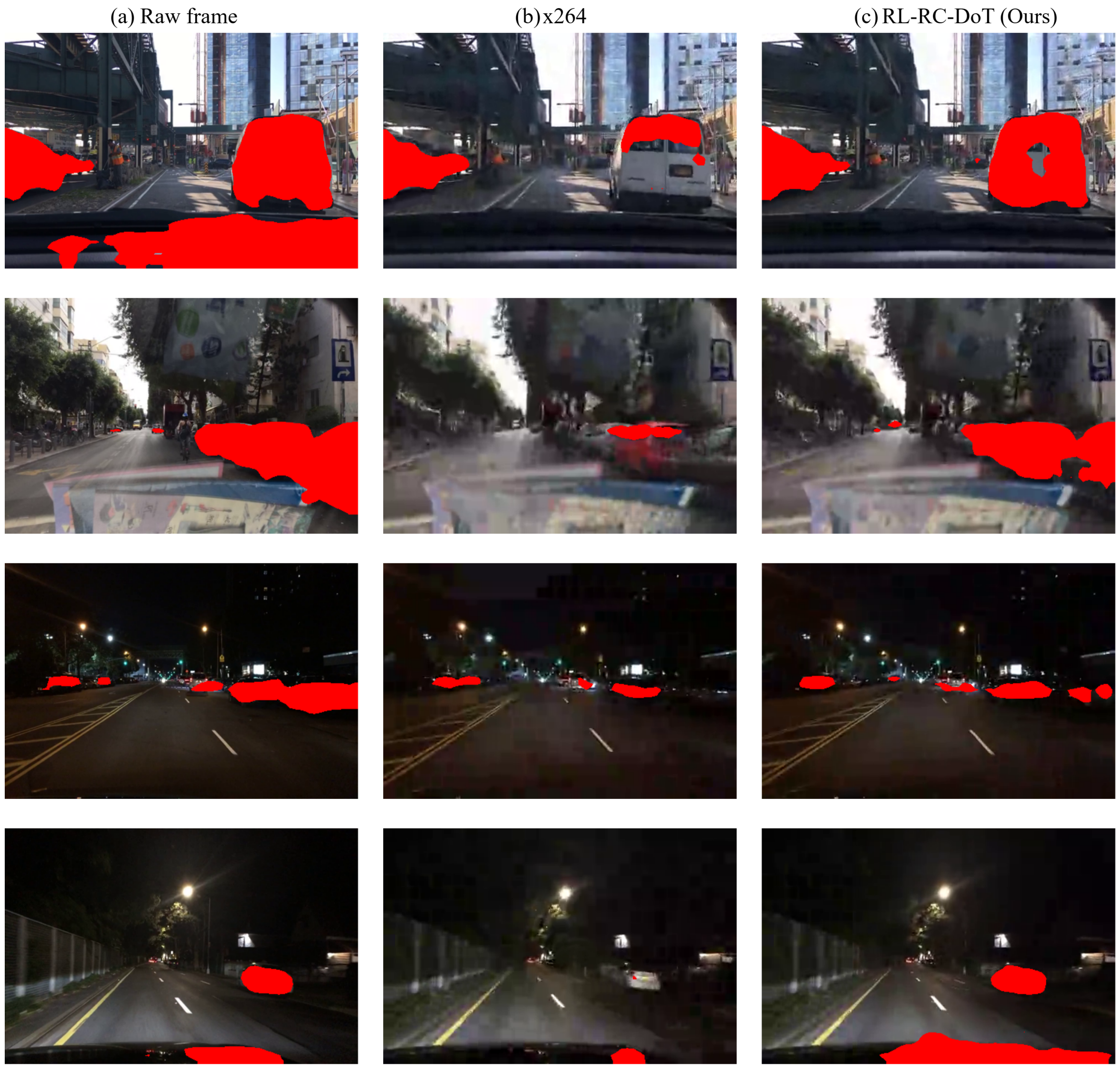

Figure 10. Car segmentation result comparison. (a) segmentation output on x264 reconstructed frame, (b) output on raw frame and (c) output on RL-RC-DoT reconstructed frame. Notice that both RL-RC-DoT and x264 used the same target bit-rate

7.3.3. Quantization map analysis

To gain more insight into how our approach affects QP map, we quantified the relation between the QP map and areas in the video that are useful for car detection. Specifically, we computed the KL-divergence between normalized QP-maps and Eigen-CAM [27] spatial map. Figure 4 illustrates the three maps for one frame, showing that RL-RC-DoT allocates more bits to areas in the frame that are informative for car detection. Figure 11 quantifies this effect across our full test data (mean D_{kl} RL-RC-DoT = 2.6 , x264=4.4).

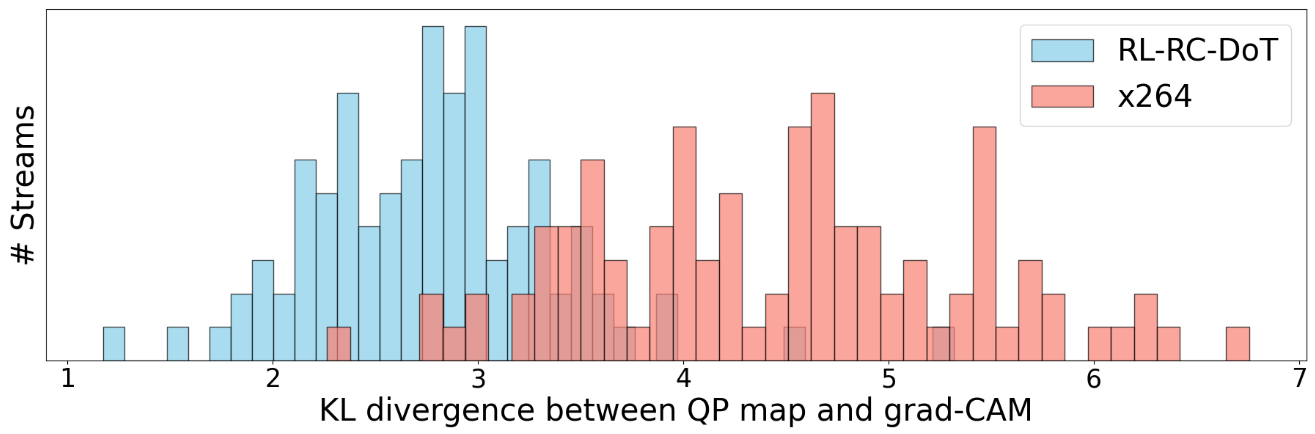

Figure 11. Distribution of average KL divergence between QP maps and Eigen-CAM.

7.4. Task accuracy to distortion trade-off

As previously discussed, RL-RC-DoT gains BD-rate reductions of $24.7\%(\pm 1.38\%)$ with respect to car detection precision task, while paying a minimal cost to overall video quality, as evidenced by a slight increase in PSNR BD-rate of $1.19\%(0.46\%)$. This is important since we want video to still be watchable by human eyes, for validation purposes and robustness to changing task models.

To further illustrate this point, in Figure 12 we show the PSNR and task performance BD-rate obtained by RL-RC-DoT for each stream in the test set. In the plots we see the PSNR varies around 0 while the tasks performance is well below.

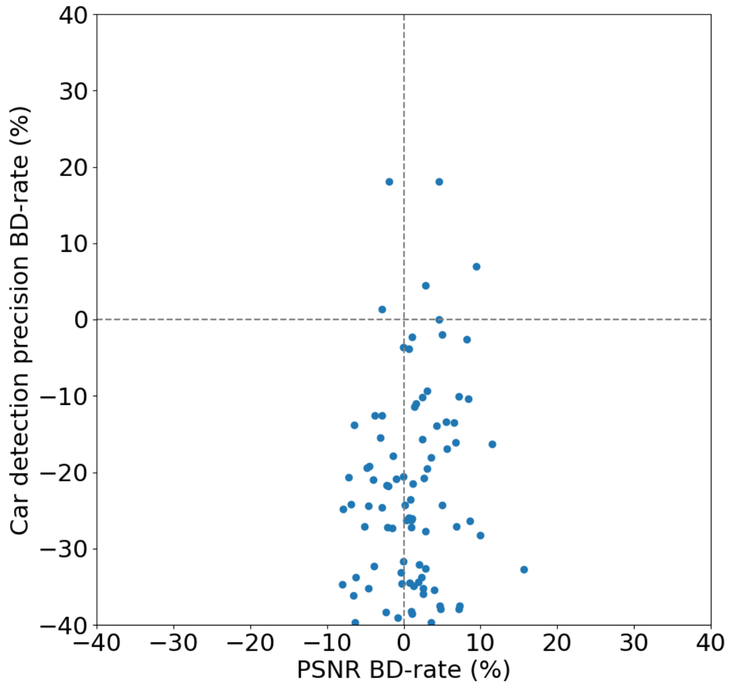

Figure 12. PSNR BD-rate to detection precision BD-rate, where each point represent a single stream in the test set

7.5. Action Space Resolution

Since we show our results on a videos of size 480x320 with macro-blocks of size 16x16, the action space is of size 30x20. The size of the action space drastically affects the performance of the agent and the convergence rate of the training process. Thus, we propose to set a lower resolution action space and upsample to the original action space by interpolation. The trade-off here is clear – if we make decisions in high resolution, the agent can take a long time to converge, whereas a low resolution decision will not provide the finer control required for accurate bit allocation for the downstream-task resulting in a sub-optimal performance. We illustrate this notion in Figure 13. We plot the task BD-rate for multiple choices of resolution reduction ratios for each of the tasks. The plot indeed shows the trade-off between the two, where each task has a different optimal choice for action space resolution. We note that these results may depend on the number of frames allotted for training, where we expect longer training to benefit lower resolutions.

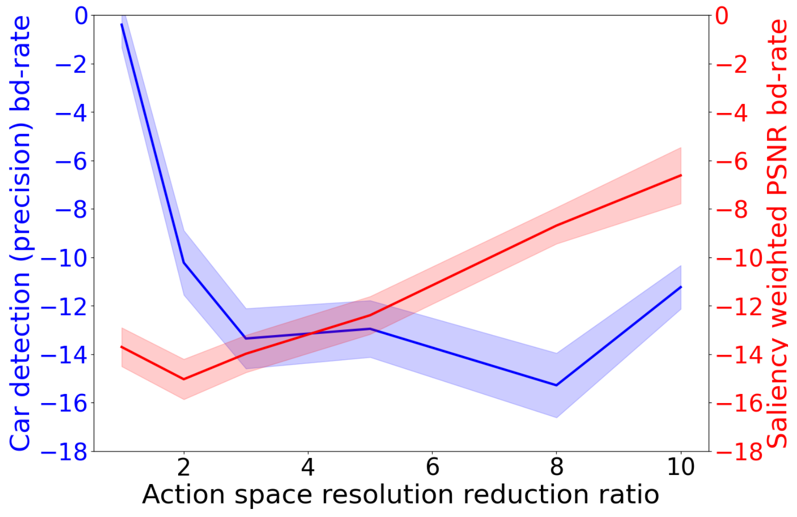

Figure 13. The effect of action space resolution on the BD-rate for both tasks

7.6. BDD100K streams

Here we elaborate on the streams we used from bdd100k dataset [46]:

Train streams	
0000f77c-6257be58	000e0252-8523a4a9
000f157f-dab3a407	000f8d37-d4c09a0f
00a04f65-af2ab984	00a0f008-3c67908e
00a0f008-a315437f	00a1176f-0652080e
00a1176f-5121b501	00a2e3ca-5c856cde
00a2e3ca-62992459	00a2f5b6-d4217a96
00a395fe-d60c0b47	00a9cd6b-b39be004
00abd8a7-ecd6fc56	00abf44e-04004ca0
00adbb3f-7757d4ea	00afa5b2-c14a542f
00afa6b9-4efe0141	00b04b30-501822fa
00b1dfed-a89dbe2b	00be7020-457a6db4
00beeb02-ba0790aa	00c12bd0-bb46e479
00c29c52-f9524f1e	00c41a61-4ba25ad4
00c497ae-595d361b	00c87627-b7f6f46c
00ca8821-db8033d5	00cb28b9-08a22af7
00ccf2e8-59a6bfc9	00ccf2e8-ac055be6
00ccf2e8-f8c69860	00ce6f6d-50bbee62
00ce8219-12c6d905	00ce8219-d0b5582e
00cef86b-204ea619	00cef86b-d8d105b9
00cf8e3d-3d27efb0	00cf8e3d-4683d983
00cf8e3d-773de15e	00cf8e3d-a7b4978c
00d0f034-6d666f7b	00d18b13-52d3e4c4
00d4b6b7-7d0a60bf	00d4b6b7-a0b1a3e0
00d7268f-fd4487be	00d79c0a-23bea078
00d79c0a-a2b85ca4	00d84b1d-21e6fe01
00d8944b-e157478b	00d8d95a-74aa476a
00d9e313-7d75bb18	00d9e313-926b6698
00dc5030-237e7f71	00de601c-858a8a8d
00de601c-cfa2404b	00e49ed1-9d41220c
00e4cae5-c0582574	00e5e793-f94de032
00e81dcc-b1dd9e7b	00e8c106-e197c4b1
00c50078-6298b9c1	00b93c6e-6298aa25
0000f77c-cb820c98	

Table 10. List of streams used in training

Validation streams	
00d8d95a-47d98291	00e02d60-54df99d1
00a820ef-d655700e	00ce95b0-84be34a3
00d15d58-9197cde54	00b04b12-a7d7eb85
00c17a92-d4803287	

Table 11. List of streams used in validation

Test streams			
cd35ea13-f49ee278	cd389564-8be2128e	cdc05b0a-3bb83a9c	cd389564-9053f5fc
cd3b1173-63cb9e2e	cd3dab20-1b3e564e	cd3dab20-4ea3d971	cd3df92f-d04e142c
cd40cb21-18170d03	cd4ac25c-61a9eb11	cd4bf816-2abb75c9	cd4bf816-c2f9bf78
cd4ce4e5-6994fd2d	cd4ce4e5-d0968ec0	cd4da443-da4fe8c7	cd4deee2-0703d1c7
cd4deee2-1d9539bd	cd4deee2-37c8b95c	cd4deee2-3feadd6e	cd4deee2-60291439
cd4deee2-688c8bba	cd4deee2-8e12e5b5	cd4deee2-9c9f6da1	cd4deee2-adc7e92a
cd4deee2-ce4f69f5	cd4deee2-d078d54a	cd547736-3b63cb96	cd583365-462cca17
cd5a94cf-345f214a	cd5a9e1b-86faac85	cd5b2540-465c9328	cd5b2540-913cb8f7
cd5bee17-bef4f177	cd5db4e0-1189ff83	cd6af452-e54a1e36	cd6c087e-03ca2127
cd6fdd33-ac9cb2db	cd704168-1231930e	cd7c12c7-7029da5d	cd7c12c7-9b46c2a8
cd7c92a7-3b20257f	cd7c92a7-89b23268	cd7c92a7-9222ee19	cd7c92a7-ed0d3926
cd7ee0b1-dd286a1b	cd7fb8f1-3d347a66	cd828461-db8b4612	cd839842-cd859db0
cd8b00aa-4aac0701	cd8b00aa-5c017145	cd8b00aa-f00ad3b9	cd8b30b0-51369077
cd8b30b0-e8d12cc4	cd8d2fde-2d2a3211	cd9b6b86-9f62a970	cd9b6b86-be582832
cd9cd3dd-d67bf5b6	cd9d84d4-f59d3feb	cd9dff27-94731aba	cd9e7e2b-4b274850
cda33556-28510da1	cda33556-8dc294b4	cda33556-c6b3dd45	cda55704-362ddfea
cda55704-754aac99	cda63e8d-0afbf52b	cda63e8d-76b2fa43	cda9acc1-1a92349d
cda9acc1-4469e473	cda9acc1-9d1ef61a	cdac4037-afed765d	cdac7315-fe37a1d9
cdae6e60-0fb06a75	cdae6e60-334ffc87	cdae6e60-b729f2e6	cdaee377-1eccb13a
cdaee377-2263611a	cdaee377-2b38ae2c	cdb06fa9-cfb70e11	cdb06fa9-eba5643a
cdb3b01b-673f85b7	cdb616df-393f382c	cdb688d4-33f24ca3	cdb6b049-c96359c8
cdb815da-d03b9395	cdb992be-f0f1613c	cdbb20a9-bdab1f4e	cdbbac37-49c0a335
cdbc7842-b72c4915	cdbd1882-bdd416ea	cdbeedfd-4ab64af8	cdbf4bd1-0c65ed7a
cdc05b0a-3bb83a9c	cdc05b0a-c53c36a6	cdc05b0a-c6e8b6ec	cdc05b0a-ce908cf7
cdc05b0a-d4ff800b	cd3dab20-1b3e564e	cdc05b0a-efb78be5	cdc05b0a-f2a67b44

Table 12. List of streams used in test

7.7. Reproducibility of experiments

Encoder environment: To apply rate-control on the environment we changed the code of the open source x264 [26] encoder so that in each frame it can obtain delta-QP values externally and provide relevant statistics as described in Appendix 7.1.1.

RL Agent: We provide a description of the policy's architecture in Appendix 7.1.2. The agent was trained using PPO implementation from stable-baselines3 [30] with default parameters, where we just added an MSE prediction loss (with weight 0.1) for reward info. We used $\lambda = 20$ to average between the bit-rate and downstream task rewards.

Experiments: In our experiments we used the publicly available BDD100K dataset (4.1) which was resized using the open source package ffmpeg[37]. We provide the named list of streams we used in Appendix 7.6. In the experimental details subsection 4.4 we provide additional information on the hardware we used and the downstream task models we used for our experiments.

Hardware: To facilitate efficient training, we utilize 8 parallel environments running on an Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, complemented by an NVIDIA Tesla V100 32GB GPU. Each agent undergoes training on 20 million frames, a process that spans approximately 4 days.