Deep Video Codec Control for Vision Models

Supplementary Material

Supplement

In this supplement, we provide additional details of our deep video codec control and conditional surrogate model. In particular, we present additional material on our methodology and also provide additional experimental results.

A. Method

This section provides additional details about our novel self-supervised control training, the surrogate model pre-training, the conditional group normalization layer, and the encoder/decoder residual block used in our conditional surrogate model. Finally, we provide a more detailed discussion of the novelty of our conditional surrogate model supporting the main paper's related work section.

A.1. Self-supervised deep codec control

Here we provide our self-supervised control training in PyTorch-like pseudo-code (Algorithm 1). Note, we train our codec control and conditional surrogate model in an alternating fashion. We utilize the prediction on the original uncompressed clip as a pseudo label in our deep codec control training (line 14). More advanced pseudo-labeling approaches (*e.g.*, [1] or [13]) might offer an improved learning signal. However, for the sake of simplicity, we refrain from using advanced pseudo-labeling approaches.

Algorithm 1 Our end-to-end deep codec control training.

```
1  for clip in data_loader:
2     # Sample BW condition
3     bw_c = log_uniform(bw_min, bw_max)
4     # Make one-hot QP prediction
5     qp_oh = control_network(clip, bw_c)
6     # Forward pass surrogate model
7     clip_c, fs = h264_sg(clip, qp_oh)
8     # Convert file size to BW
9     bw = file_size_to_bandwidth(fs)
10    # Make downstream prediction
11    pred = downstream_model(clip_c)
12    # Downstream prediction raw clip
13    with no_grad():
14       label_pseudo = downstream_model(clip)
15    # Compute loss
16    loss = a_b * l_b(bw, bw_c) \
17       + a_r * regularizer_b(bw, bw_c) \
18       + a_p * h(bw_c - bw) * l_p(pred, label_pseudo)
19    # Backward pass
20    loss.backward()
21    # Optimization step
22    optimizer_control_network.step()
23    # Update EMA control network
24    ema(control_network, control_network_ema, decay=0.99)
25    # Next: Surrogate model training step
```

A.2. Surrogate model pre-training

During preliminary surrogate model pre-training runs, we observed that the surrogate model entails a tendency to be biased toward the identity function. In particular, the surrogate model solved the task of predicting the distorted video by predicting the original video. We combat this behavior by first learning on high quantization parameter (QP) values and gradually introducing a larger range of QP during training. We start by just sampling a macroblock-wise QP of 51 before linearly increasing the QP sampling range to the full range between 0 and 51 until half of the pre-training. We observed that first learning strong compression rates, including strong video distortion, prevents the surrogate from just learning the identity mapping. We believe the strong video distortion at the beginning of the training lets the conditional surrogate model diverge from the shortcut solution (identity mapping).

To artificially enlarge the number of available video clips, we use data augmentation during surrogate pre-training. In particular, we randomly (with a probability of 0.1) convert the RGB frames to grayscale frames. We also utilize temporal augmentations. We randomly reverse the order of frames (with a probability of 0.5) and repeat frames (with a probability of 0.1).

We sample the QP parameters at different resolution stages to mimic regions with uniform QP values. Additionally, with a probability of 0.4, we utilize the same QP map for all frames in the video clip, mimicking uniform QP values in the spatial dimension. QP sample generated by our technique can be seen in Fig. 15 (and subsequent Figures).

A.3. Conditional group normalization

To incorporate the one-hot macroblock-wise quantization parameters $\mathbf{qp}$ into our conditional surrogate model, we utilize conditional group normalization (CGN). Before applying CGN, we embed $\mathbf{qp}$ using a two-layer feed-forward neural network to the latent vector $\mathbf{z}$. Similar to conditional batch normalization [5], we predict the affine parameters of group normalization [31] based on the condition latent vector $\mathbf{z}$. In particular, we use the spatial feature transform layer introduced by Wang *et al*. [29] to predict macroblock-wise affine parameters. Formally, our CGN later is described as

$$\hat{\mathbf{X}} = \mathrm{Softplus}\big(\xi_\mu(\mathbf{z})\big)\,\mathrm{GroupNorm}(\mathbf{X}) + \xi_\sigma(\mathbf{z})\,, \tag{1}$$

where $\mathbf{X}$ is the 4D spatio-temporal input feature map and $\hat{\mathbf{X}}$ is the output feature map of the same shape. $\mathbf{z}$ is the 4D condition embedding generated from $\mathbf{qp}$. ξ denotes a learnable linear mapping transforming the condition embedding. A Softplus activation [19] is used to ensure a positive scaling. $\mathrm{GroupNorm}$ denotes the standard group normalization layer without affine parameters. To ensure matching spatial dimensions between the feature map and macroblock-wise affine parameters, nearest-neighbor interpolation is used to the output of ξ_μ and ξ_σ.

A.4. Encoder/decoder 2D residual block

Both our surrogate model encoder and decoder utilize 2D residual building blocks [11]. In particular, our residual block is composed of two 2D 3×3 convolutions, two leaky ReLU activations [17], a CGN layer, and a standard GN layer [31]. In the encoder, we utilize a strided convolution to reduce the spatial dimensions. To upsample the feature maps in the decoder, we employ a 4×4 transposed convolution instead of the first 3×3 convolution. The full block is visualized in Fig. 1.

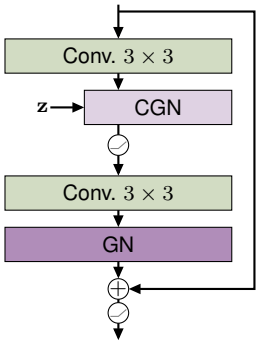

Figure 1. Residual encoder and decoder block conditioned on the latent vector $\mathbf{z}$.

A.5. Conditional surrogate model discussion.

Note that we are not the first to propose a differentiable surrogate model of a non-differentiable standard video codec (*e.g.*, H.264 or H.265) [12, 14, 22, 25, 33]. However, to the best of our knowledge, we are proposing the first surrogate model offering support for fine-grain conditioning (macroblock-wise quantization). Additionally, our conditional surrogate model offers a differentiable prediction of the file size of the encoded video. While we are the first to offer a differentiable file size prediction for a standard video codec, existing work on surrogate modeling image codecs (*e.g.*, JPEG [28]) also offers differentiable file size predictions [16, 32]. In summary, our novel conditional surrogate model is the first to approximate the H.264 standard video codec while offering macroblock-wise conditioning (over the whole QP range) and a differentiable file size prediction. This conditional surrogate model enables us the learn our deep video codec control in a fully end-to-end fashion, while existing surrogate models are unable to facilitate the end-to-end learning.

B. Experiments

First, we provide additional implementation details, before introducing our downstream models and their downstream performance. Next, we showcase additional results on how H.264 coding affects the downstream performance of deep vision models. Finally, we provide additional deep codec control and surrogate model results.

B.1. Further implementation details

We implement our surrogate model and codec control pipeline using PyTorch [21], PyTorch Lightning [7], and Kornia [23]. For the macroblock-wise H.264 compression, we rely on the modified FFmpeg implementation of AccMPEG [6, 26]. We pre-train our conditional surrogate model for 45k iterations with a base learning rate of $4 \cdot 10^{-4}$ and a weight decay of 10^{-5}. We employ a cosine learning rate schedule with linear annealing [15]. Our deep codec control is trained for just 4.5k iterations. For training the control network, we also use the AdamW optimizer with a cosine learning rate schedule (without annealing). The base learning rates are set to 10^{-4} for the prediction head and to 10^{-5} for the pre-trained backbone blocks of the control network. We use a weight decay of 10^{-3}. For fine-tuning the conditional surrogate model, we use a fixed learning rate of $1 \cdot 10^{-4}$. For both surrogate pre-training and codec control training, we utilize two NVIDIA A6000 (48GB) GPUs. Surrogate pre-training takes approximately one day, whereas our codec control training requires approximately 12 hours to complete. For both surrogate pre-training and codec control training, we use a batch size of 8 per GPU and half-precision training.

The surrogate model's encoder uses four residual blocks with 64, 128, 256, and 1024 convolutional filters, respectively. The decoder is composed of four residual blocks with 512, 256, 128, and 64 convolutional filters, respectively. A conditional embedding dimension $\text{C}_\mathbf{z}$ of 256 is used. The AGRU utilizes a channel dimension of 1024 for each convolution and eight recurrent iterations. For computing the optical flow used to align features before each AGRU iteration, we use a pre-trained RAFT small model from TorchVision [24, 27]. We train the RAFT weights together with the other surrogate parameters. For the backbone blocks (X3D-S [9]) of the control network, we utilize pre-trained weights obtained from action recognition on Kinetics-400 [8]. We freeze the backbones batch normalization layer during training. For the prediction head, we use two conditional 3D residual blocks (*cf*. Fig. 2). During codec control training we start with a Gumbel-Softmax temperature of 2.0 and decrease the temperature in a cosine schedule to 0.1 at the end of the training. In total, the control network is composed of only 3M parameters, making it easier to deploy the control network on an edge device, such as the NVIDIA Jetson Nano.

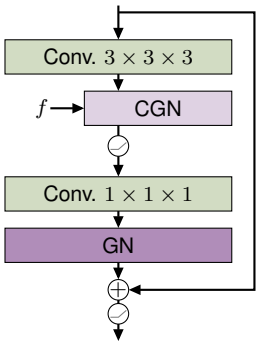

Figure 2. Conditional 3D residual block composed of two convolutions, two leaky ReLU activations, a GN layer, and a CGN layer. For the first convolution, we use a group size equal to the number of channels.

As the segmentation model, we utilize our trained DeepLabV3 models (Cityscapes & CamVid) trained with the MMSegmentation [18] protocol and H.264 augmentation. For optical flow estimation, we utilize a RAFT large model from Torchvision [24, 27].

We utilize both Cityscapes [4] and CamVid [2] with a resolution of 224×224. On Cityscapes, this resolution is achieved by downsampling the frame by a factor of 4 followed by cropping. On CamVid, we downsample the frames by a factor of 3 followed by clipping. We use a resolution of 224×224 to combat the large memory requirements of our codec control pipeline introduced by our surrogate model.

We set the surrogate loss weights to $\alpha_{\rho_v} = 10^{-4}$, $\alpha_{\text{SSIM}} = 2$, $\alpha_{\text{FF}} = 200$, $\alpha_{\rho_f} = 10^{-4}$, and $\alpha_{\text{L1}} = 0.1$. We use $\alpha_{\text{p}} = 2$, $\alpha_{\text{b}} = 6$, $\alpha_{\text{r}} = 1$, $\epsilon_{\text{b}} = \epsilon_{\text{p}} = 0.02$, and $\epsilon_{\text{r}} = 0.05$ for the control loss. We sample the bandwidth condition $\tilde{b}$ from a log uniform distribution $\mathcal{U}_{\log}(30\text{kbit/s}, 0.9\text{Mbit/s})$ capturing the working range of the H.264 codec (*cf*. Fig. 3). We use a log-uniform distribution since the bandwidth correlates logarithmically with QP and we want to explore QP uniformly during training. For validation, we sample 10 bandwidth conditions equally spaced in $\log_{10}$-space.

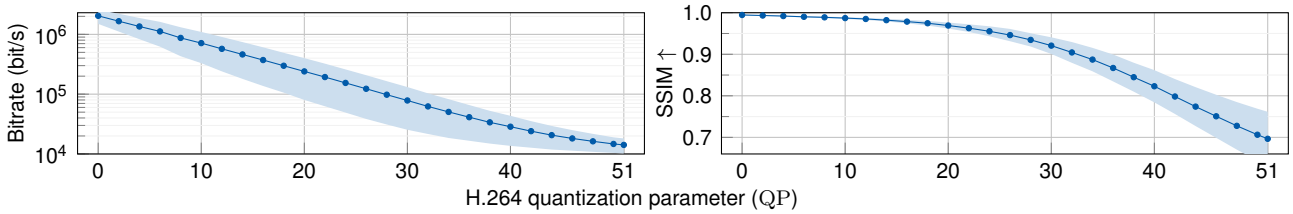

Figure 3. **Rate-distortion tradeoff.** Bandwidth and SSIM (to raw video) with two standard dev. for different quantization parameters (QP). A uniform QP over all macroblocks is used. Cityscapes (val seq.) with a resolution of 224×224, a clip length (and GOP) of 8, and a temporal stride of 3 is utilized. Frame rate is 17fps.

B.2. Downstream models

For our semantic segmentation codec control experiments, we utilize a DeepLabV3 [3] with ResNet-18 [11] backbone (w/ H.264 aug.). We trained this model on the resolution (224×224) used by our codec control. Additionally, we also trained different variants of the model for analyzing the downstream performance of these models on H.264 codec video. To showcase the effect of H.264 as a data augmentation during downstream training, we train each model variant with and without H.264 as data augmentation. In our H.264 data aug. implementation, we use H.264 coded frames (with random QP) in 50% of the case. Except for the optional addition of the H.264 aug., we follow the Cityscapes training protocol of MMSegmentation [18] for both the Cityscapes and the CamVid datasets. Downstream validation results on non-coded frames are shown in Tab. 1.

Table 1. **Downstream model results (semantic segmentation).** We report the mean IoU on the respective validation set using the MMSegmentation validation pipeline.

Method	H.264 aug.	mIoU (%) ↑
	Cityscapes [4]	
DeepLabV3 (ResNet-18)	✗	64.47
DeepLabV3 (ResNet-18)	✓	64.96
DeepLabV3 (ResNet-50)	✗	67.62
DeepLabV3 (ResNet-50)	✓	66.74
	CamVid [2]	
DeepLabV3 (ResNet-18)	✗	50.68
DeepLabV3 (ResNet-18)	✓	50.48
DeepLabV3 (ResNet-50)	✗	56.65
DeepLabV3 (ResNet-50)	✓	55.35

For our optical flow codec control experiments, we utilize a RAFT [24] large model, trained supervised on synthetic data and KITTI [10]. In particular, we utilize the public checkpoint from Torchvision [27].

B.3. Deep vision performance on H.264 coded videos

We analyze the performance of our downstream models trained for semantic segmentation on H.264 coded videos. We also analyze the optical flow estimation performance of RAFT (large & small) on coded videos. Cityscapes results are shown in

Fig. 4. If QP is increased downstream performance on Cityscapes is vastly deteriorated.

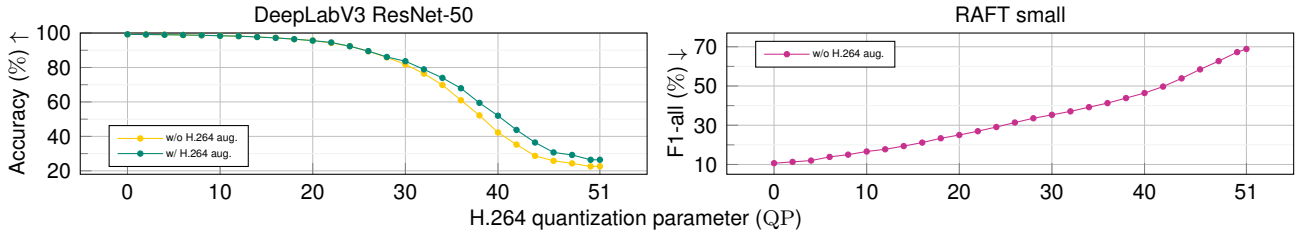

Figure 4. **Downstream performance *vs*. compression trade-off.** Cityscapes (val sequences) segmentation accuracy and optical flow estimation performance, measured by average endpoint error (AEPE), for different H.264 quantization parameters between the raw clip predictions (pseudo label) and the compressed clip predictions. Temporal stride is 3 and clip length (and GOP) is 8. QP is applied uniformly.

Downstream performance results on CamVid using H.264 coded videos are shown in Fig. 6. Consistent with the results on Cityscapes, downstream performance on CamVid also deteriorates as QP is increased. These findings also align with the findings by Otani *et al*. [20] on the deterioration of action recognition performance on H.264 (and JPEG [28]) coded videos/frames.

Using H.264 coded frames during supervised training only leads to minor residents to coded frames during inference. Still, downstream segmentation performance is highly affected. Interestingly, H.264 augmentation seems to have a better impact if the backbone network has a higher capacity.

B.4. Codec control results: semantic segmentation

We present additional qualitative and quantitative results of our deep video codec control on semantic segmentation as a downstream task. In Fig. 5, we showcase our deep codec control on a sequence of 12 video clips of the Cityscapes Stuttgart demo. Our codec control network adapts the macroblock-wise QP to the bandwidth constraints and captures interesting regions. Typically, the control network assigns lower QP values (less compression) to crowded frame regions. Still, the segmentation is affected if only limited bandwidth is available. We also observe the tendency of 2-pass ABR to overshoot the target bandwidth. In contrast, our deep codec control better follows the target bandwidth.

In Fig. 7 & Fig. 8, we provide fine-grain results to Tab. 1 of the main paper on the Cityscapes dataset. Our deep video codec control largely outperforms 2-pass ABR control. However, for some bandwidth conditions, we perform on par with 2-pass ABR. Surprisingly, we observe that 2-pass ABR not only struggles to meet the bandwidth conditioning for low bandwidth values but also tends to overshoot the target bandwidth for larger bandwidth values.

In Tab. 2, we ablate the use of EMA on the control networks parameters. While scoring similar bandwidth condition accuracies EMA helps in downstream performance. In the most relevant setting, with no bandwidth tolerance, using EMA improves $\mathrm{acc}_{\mathrm{seg}}$ by approximately a half percent.

Table 2. **EMA ablation results on Cityscapes (semantic segmentation).** Here we compare our learn control with and without EMA (exponential moving average) on the parameters. We report bandwidth ($\mathrm{acc}_{\mathrm{bw}}$) and segmentation accuracies ($\mathrm{acc}_{\mathrm{seg}}$) for difference bandwidth tolerances. Metrics averaged over ten different bandwidth conditions.

	$\mathrm{acc}_{\mathrm{bw}}$ ↑			$\mathrm{acc}_{\mathrm{seg}}$ ↑		
Control network EMA	$\Delta 0\%$	$\Delta 5\%$	$\Delta 10\%$	$\Delta 0\%$	$\Delta 5\%$	$\Delta 10\%$
✓	96.22	97.05	97.91	84.79	85.50	86.28
✗	96.12	97.01	96.89	84.36	85.15	86.12

B.5. Codec control results: optical flow estimation

In Fig. 7 & Fig. 8, we provide fine-grain results to Tab. 2 of the main paper on the CamVid dataset. Similar to semantic segmentation, our deep codec control better follows the bandwidth conditioning and also better preserves downstream performance over most of the bandwidth range. Interestingly, for large bandwidth conditions, 2-pass ABR control tends to lead to slightly better downstream results. However, in general, our deep codec control leads to superior results. Especially for low network bandwidth conditions, our deep codec control vastly outperforms 2-pass ABR control.

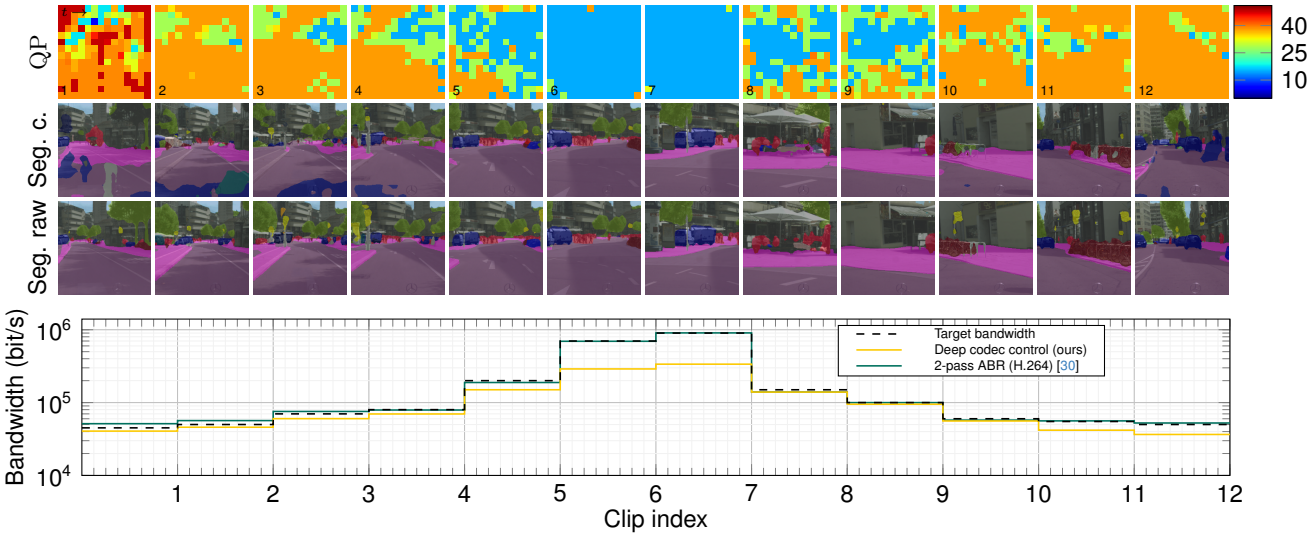

Figure 5. **Qualitative results.** Cityscapes Stuttgart demo [4] with clip-wise varying bandwidth condition. 12 clips with 8 frames each (temp. stride 3) used. QP prediction in the top row, segmentation prediction on the compressed video middle row, and segmentation prediction on the raw video bottom row. For clarity, we only visualize the QP prediction and segmentation for the first frame of each clip. DeepLabV3 (ResNet18 and H.264 aug. training) utilized. The graph shows the generated clip-wise bandwidths by H.264 and our approach. The segmentation accuracy (considering drops) over all clips is 45.71% for H.264 and 82.98% for our deep codec control. Zoom in for details; best viewed in color.

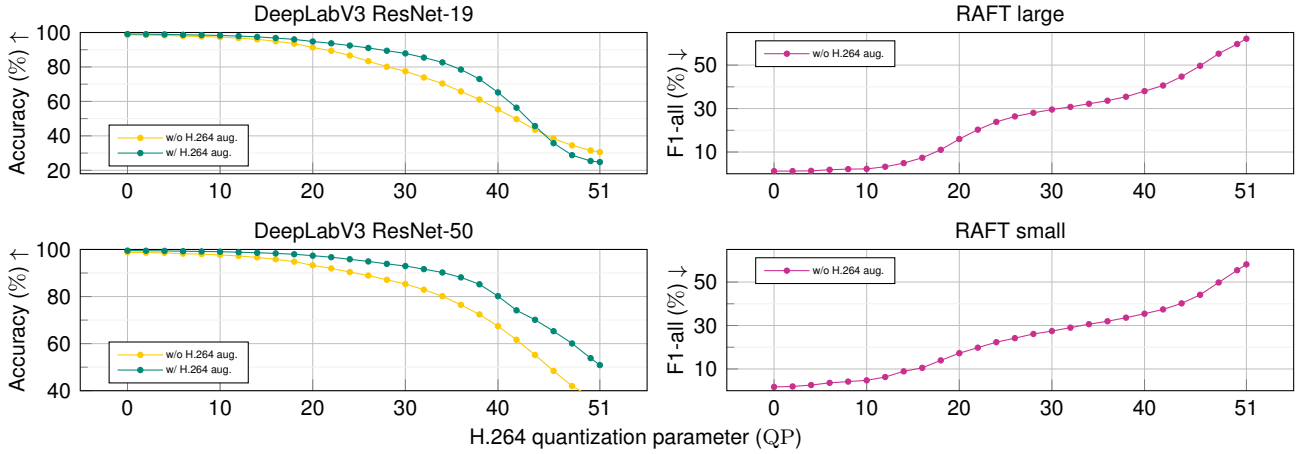

Figure 6. **Downstream performance *vs*. compression trade-off.** CamVid (val) segmentation accuracy and optical flow estimation performance, measured by average endpoint error (AEPE), for different H.264 quantization parameters between the raw clip predictions (pseudo label) and the compressed clip predictions. Temporal stride is 3 and clip length (and GOP) is 8. QP is applied uniformly.

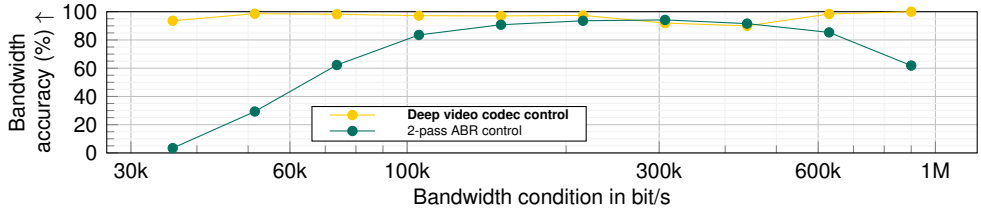

Figure 7. **Bandwidth condition results.** Bandwidth accuracy ($\mathrm{acc}_{\mathrm{bw}}$), *i.e.*, how often the bandwidth constraint has been satisfied, for different network bandwidth conditions on Cityscapes (val sequences).

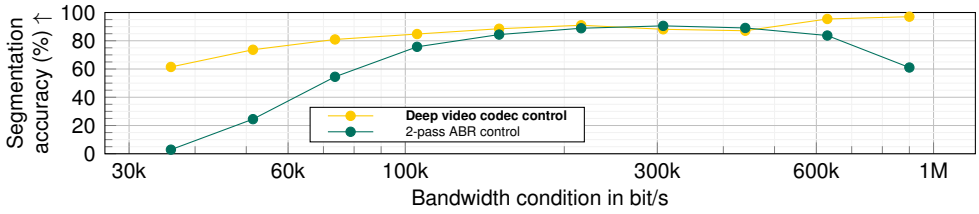

Figure 8. **Downstream results.** Segmentation accuracy ($\mathrm{acc}_{\mathrm{seg}}$), considering clip dropping, for different network bandwidth conditions on Cityscapes (val sequences).

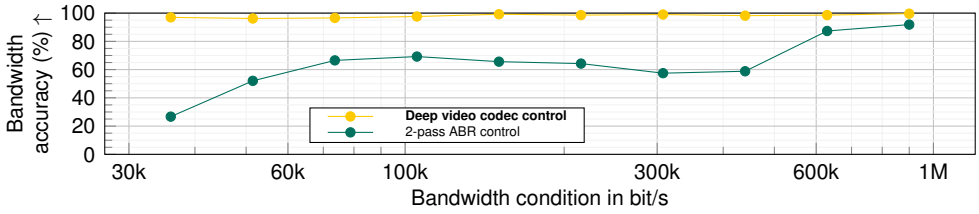

Figure 9. **Bandwidth condition results.** Bandwidth accuracy ($\mathrm{acc}_{\mathrm{bw}}$), *i.e.*, how often the bandwidth constraint has been satisfied, for different network bandwidth conditions on CamVid (val sequences).

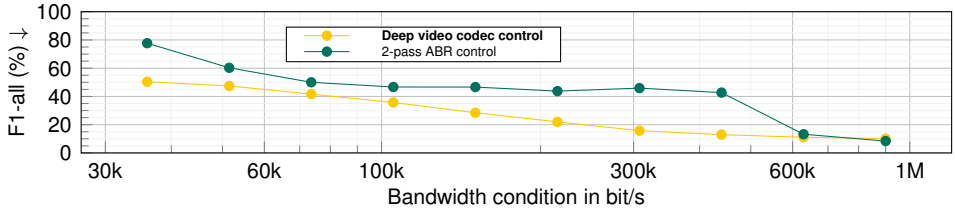

Figure 10. **Downstream results.** F1-all, considering clip dropping, for different network bandwidth conditions on CamVid (val sequences).

B.6. Surrogate model results

We provide fine-grain results of our conditional surrogate model on CamVid (Fig. 11 & Fig. 12) and Cityscapes (Fig. 13 & Fig. 14). Note our conditional surrogate model is considerably better than using the naive solution, the identity function. This can be observed by comparing Fig. 11 and Fig. 2 of the main paper.

In the following Fig. 15-Fig. 20, we provide additional qualitative results of our conditional surrogate in approximating H.264 video distortion for different macroblock-wise quantization parameters. Note we present standard video clips but also show video clips with repeated frames and in reversed order, showcasing the generalization ability of our conditional surrogate model. We observe our surrogate model is able to accurately approximate the video distortion of H.264 for different macroblock-wise quantization parameters.

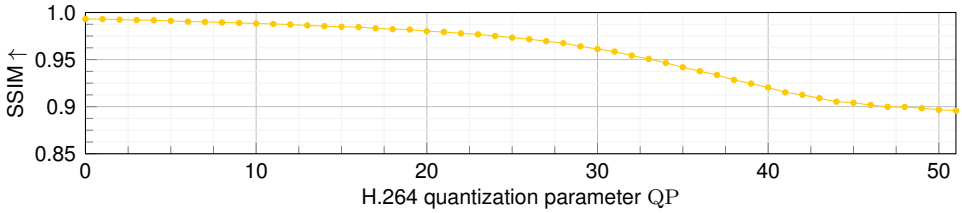

Figure 11. **Surrogate model coded video prediction results on CamVid.** We utilize a uniform QP (for all macroblocks).

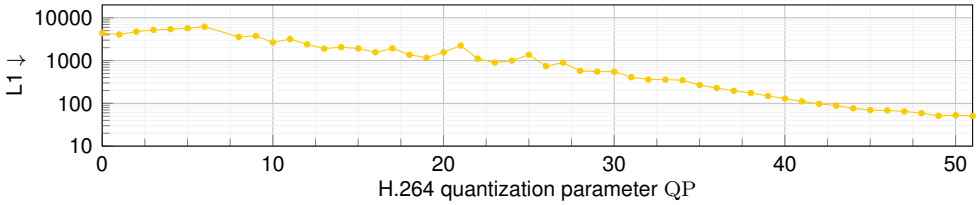

Figure 12. **Surrogate model file size prediction results on CamVid.** We utilize a uniform QP (for all macroblocks).

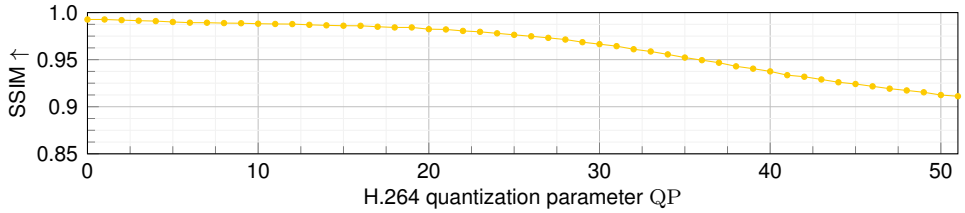

Figure 13. **Surrogate model coded video prediction results on CamVid.** We utilize a uniform QP (for all macroblocks).

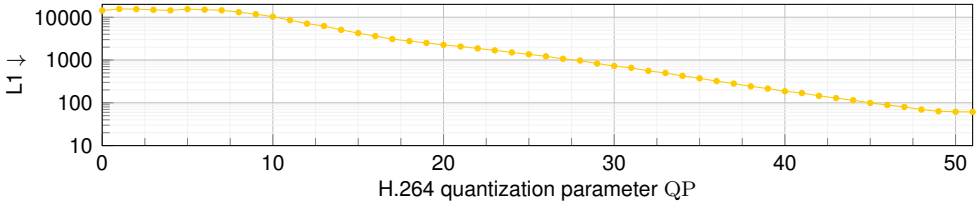

Figure 14. **Surrogate model file size prediction results on Cityscapes.** We utilize a uniform QP (for all macroblocks).

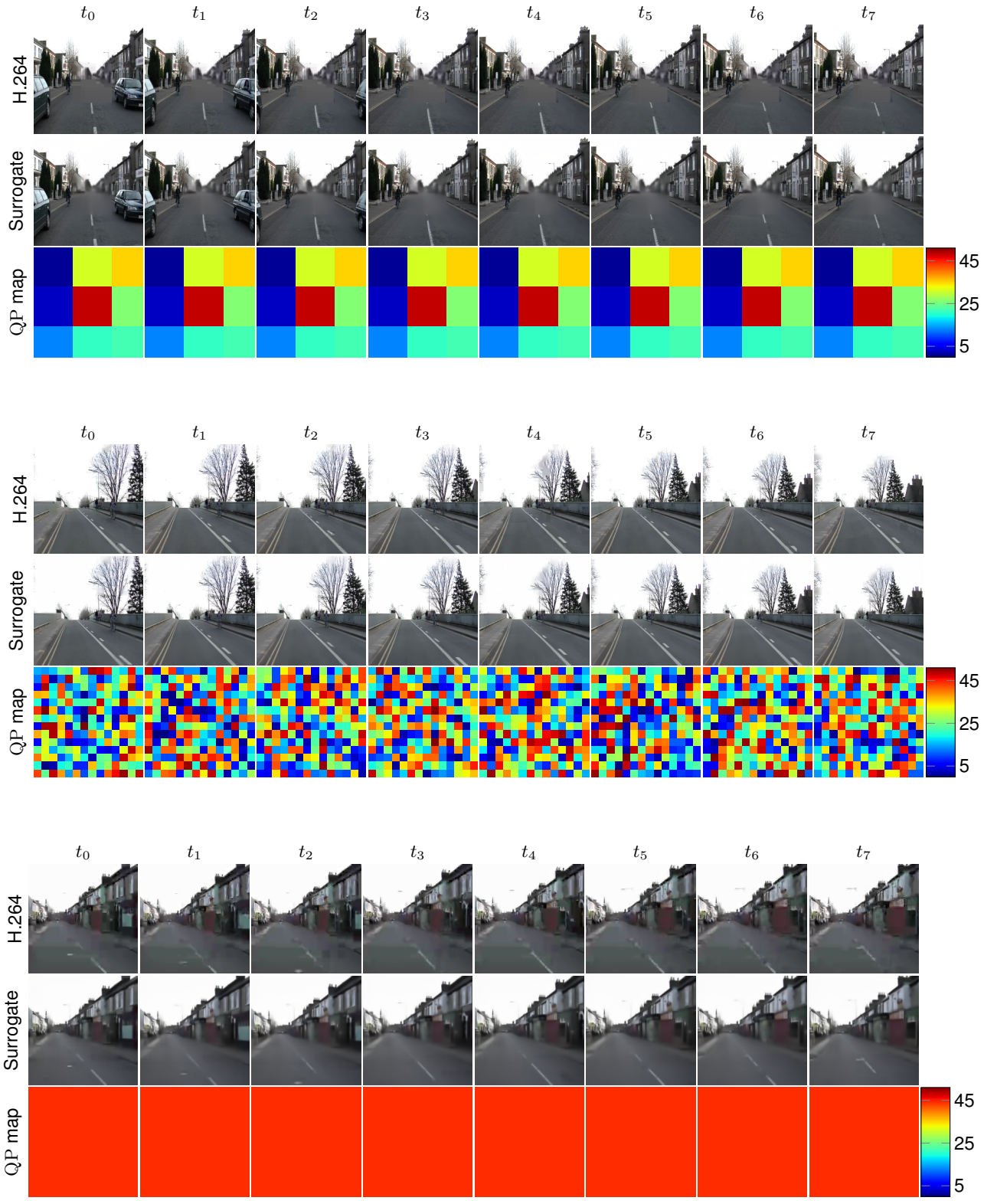

Figure 15. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled over the full range. CamVid dataset used.

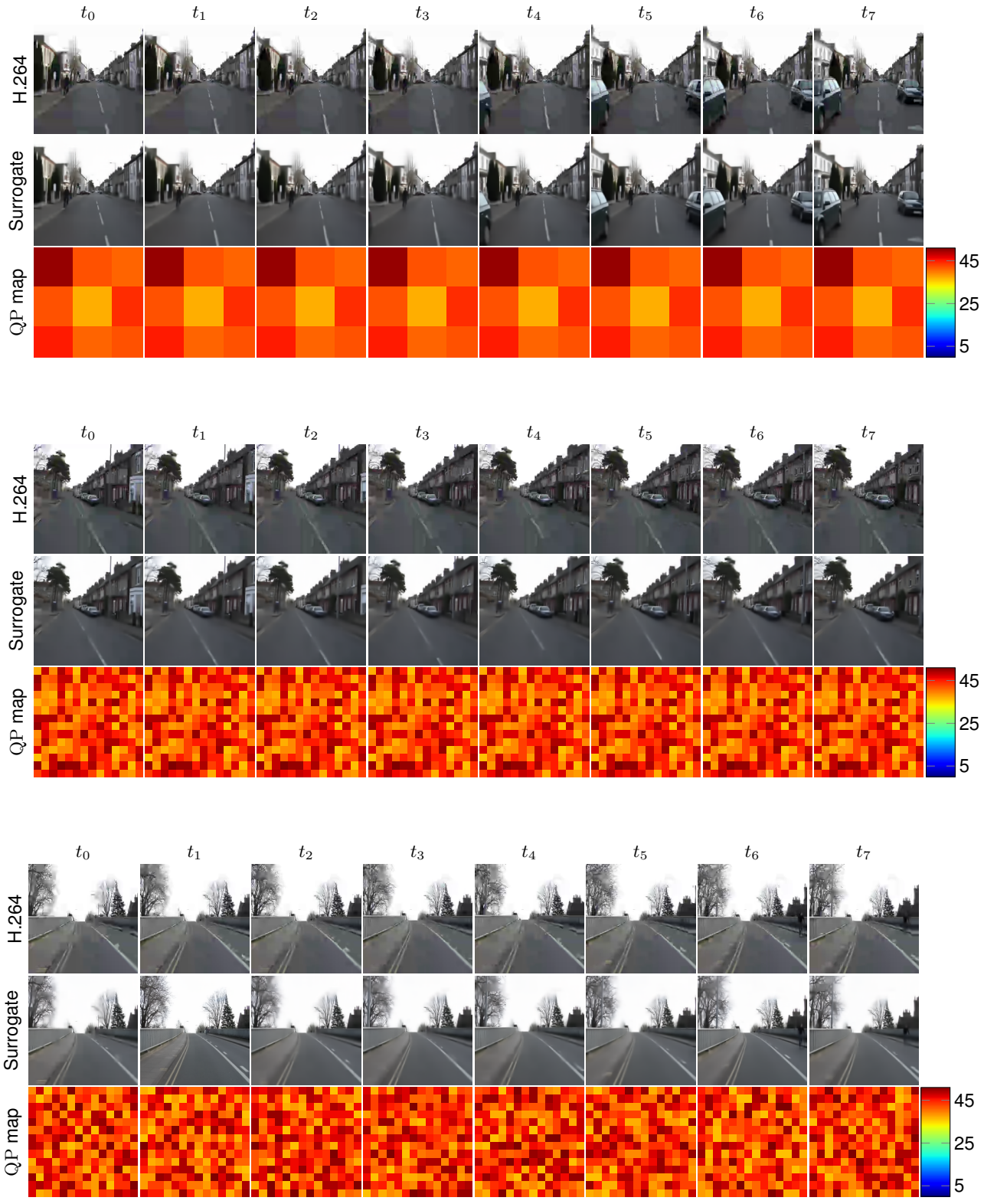

Figure 16. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled from a range between 36 and 51. CamVid dataset used.

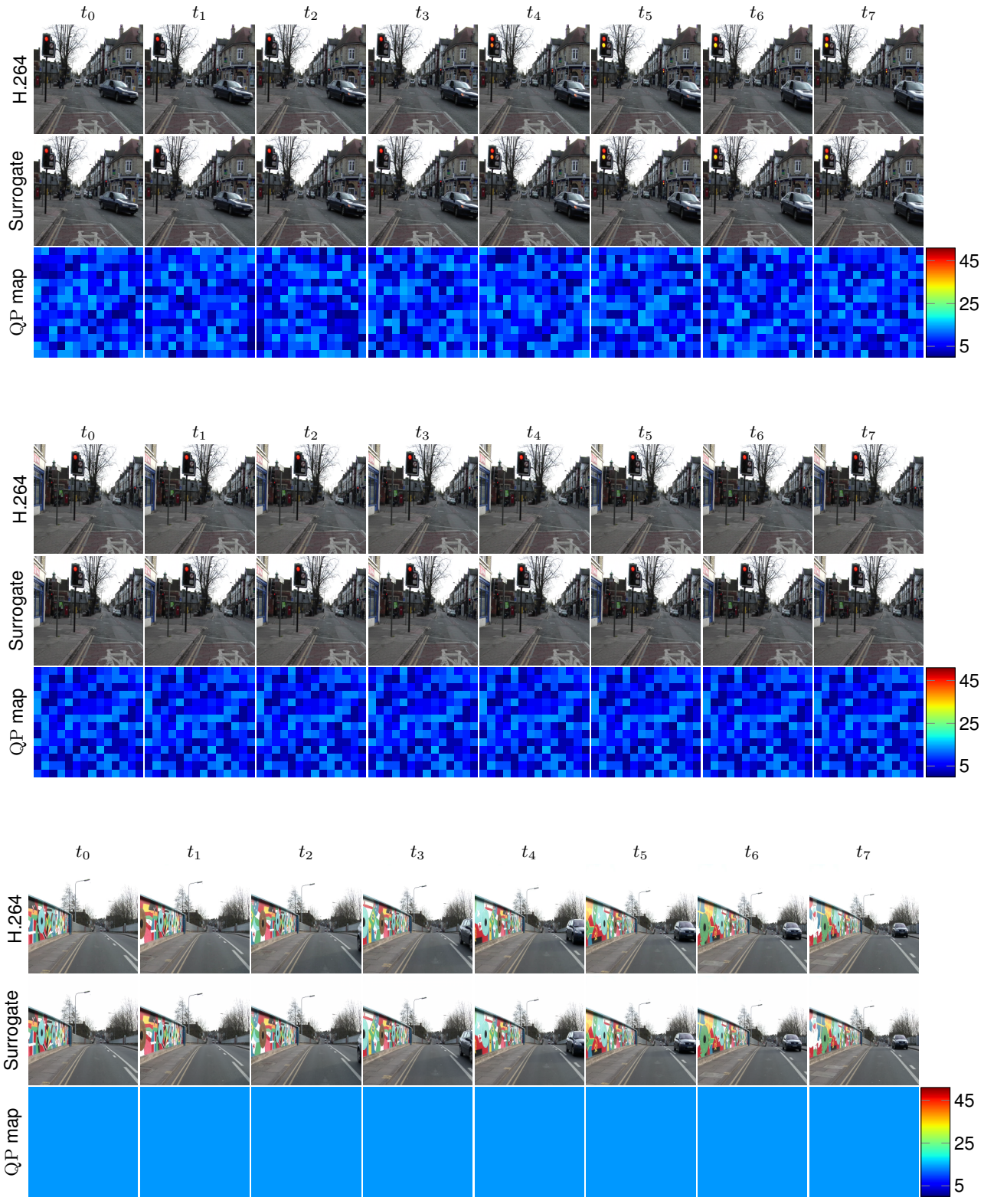

Figure 17. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled from a range between 0 and 15. CamVid dataset used.

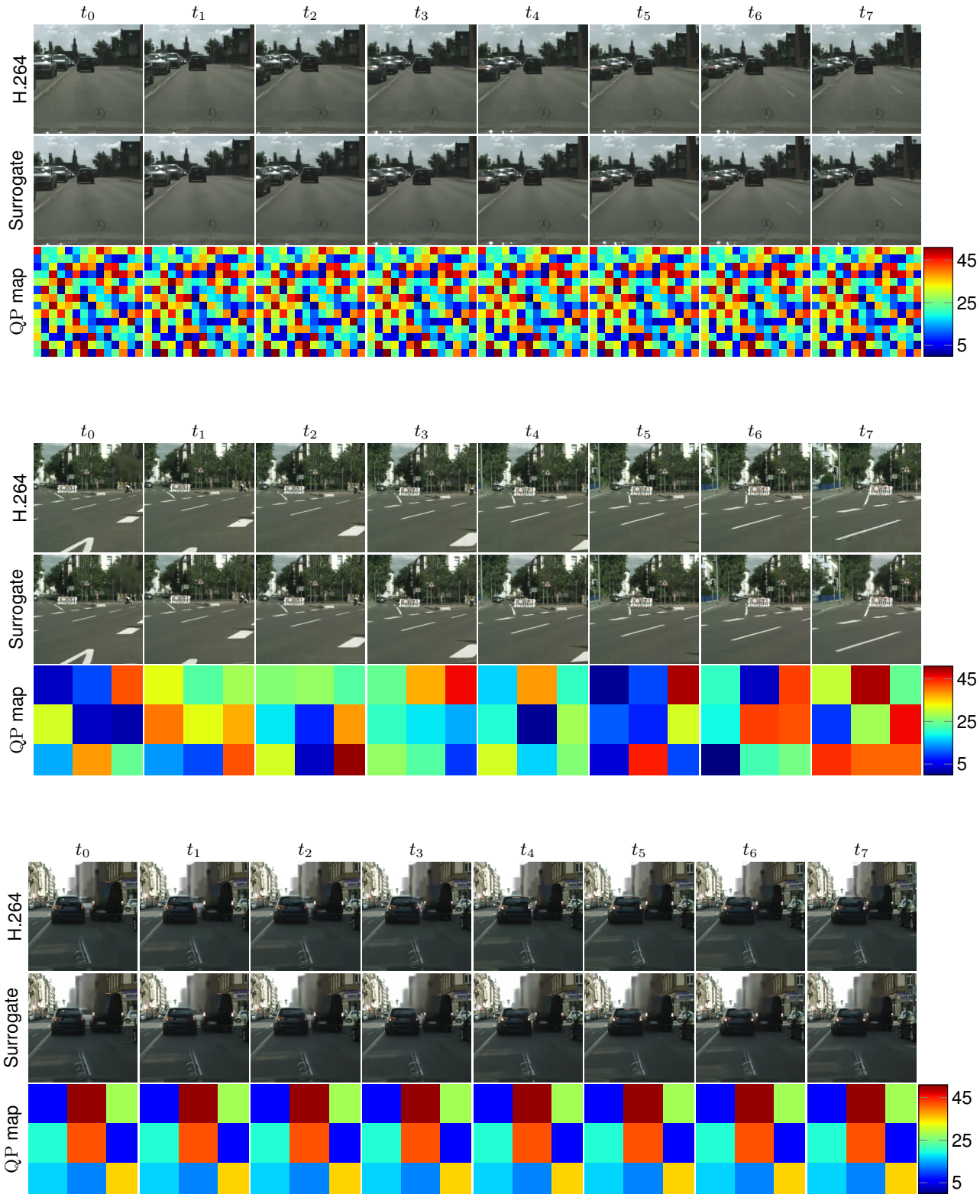

Figure 18. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled over the full range. Cityscapes dataset used.

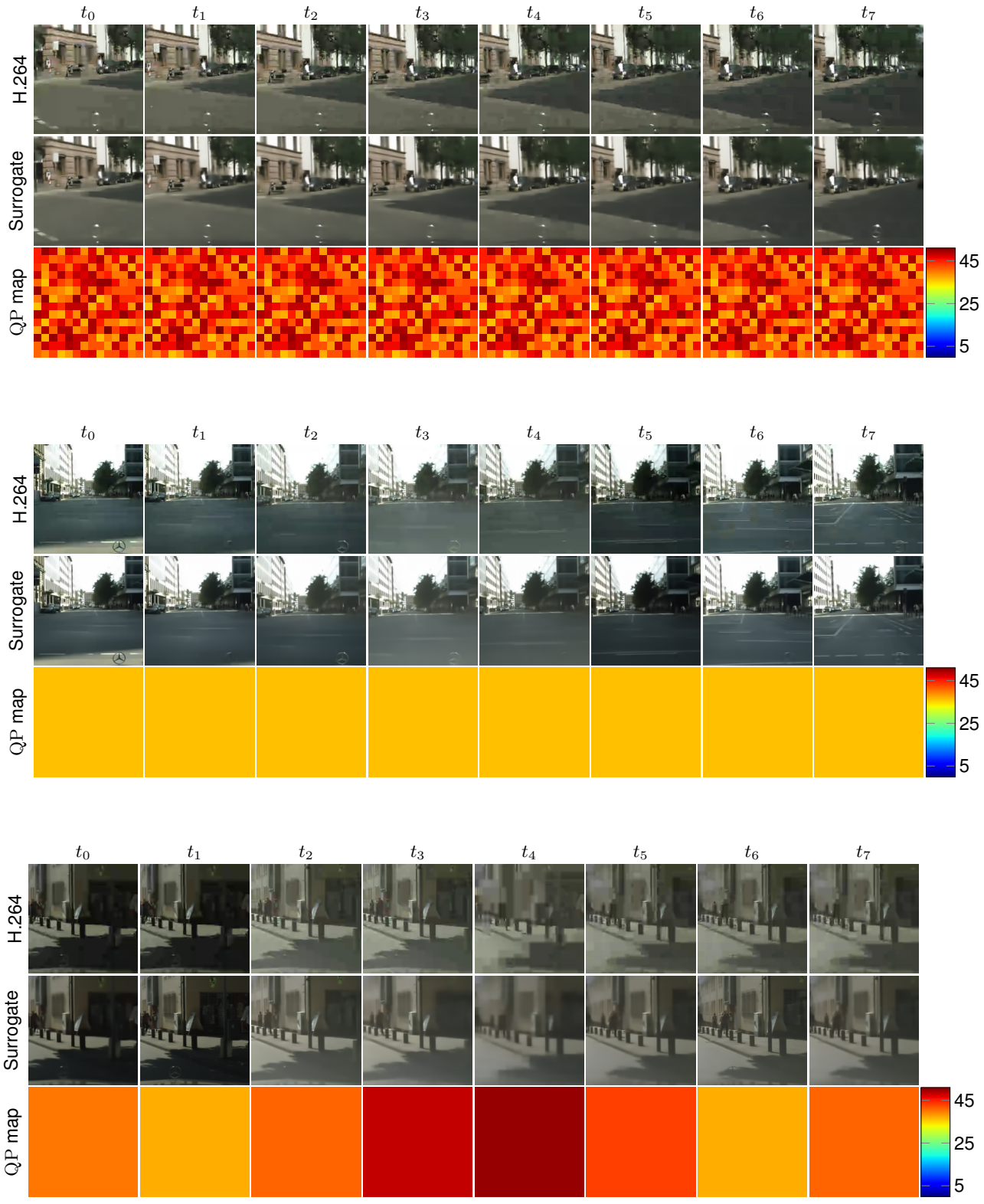

Figure 19. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled from a range between 36 and 51. Cityscapes dataset used.

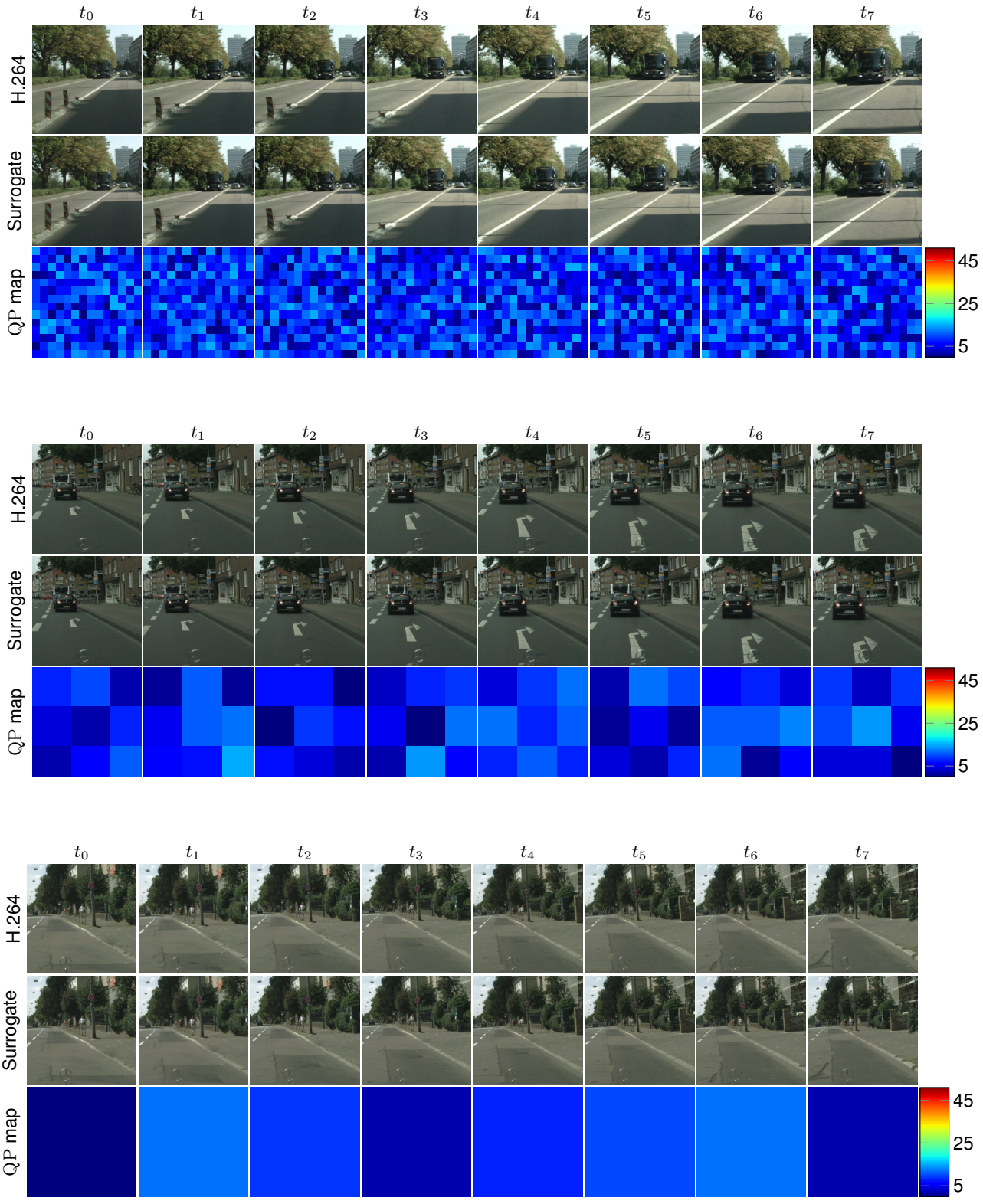

Figure 20. **Qualitative surrogate model results.** The first row shows the H.264 coded clip, the middle row shows the prediction of our conditional surrogate model, and the bottom row shows the macroblock-wise QP map. QP sampled from a range between 0 and 15. Cityscapes dataset used.

References

[1] Nikita Araslanov and Stefan Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *CVPR*, pages 15384–15394, 2021. 1

[2] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.*, 30(2):88–97, 2009. 4

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587 [cs.CV]*, 2017. 4

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 4, 6

[5] Harm de Vries, Florian Strub, Jeremie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *NIPS*, 2017. 2

[6] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. AccMPEG: Optimizing video encoding for accurate video analytics. In *MLSys*, pages 450–466, 2022. 3

[7] William Falcon and The PyTorch Lightning Team. PyTorch Lightning. https://github.com/Lightning-AI/lightning, 2019. 3

[8] Haoqi Fan, Tullie Murrell, Heng Wang, Kalyan Vasudev Alwala, Yanghao Li, Yilei Li, Bo Xiong, Nikhila Ravi, Meng Li, Haichuan Yang, et al. PyTorchVideo: A deep learning library for video understanding. In *ACMMM*, pages 3783–3786, 2021. 3

[9] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, pages 203–213, 2020. 3

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.*, 32(11):1231–1237, 2013. 4

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2, 4

[12] Berivan Isik, Onur G Guleryuz, Danhang Tang, Jonathan Taylor, and Philip A Chou. Sandwiched video compression: Efficiently extending the reach of standard codecs with neural wrappers. *arXiv:2303.11473 [eess.IV]*, 2023. 3

[13] Jisoo Jeong, Hong Cai, Risheek Garrepalli, and Fatih Porikli. DistractFlow: Improving optical flow estimation via realistic distractions and pseudo-labeling. In *CVPR*, pages 13691–13700, 2023. 1

[14] Jan P Klopp, Keng-Chi Liu, Liang-Gee Chen, and Shao-Yi Chien. How to exploit the transferability of learned image compression to conventional codecs. In *CVPR*, pages 16165–16174, 2021. 3

[15] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3

[16] Xiyang Luo, Hossein Talebi, Feng Yang, Michael Elad, and Peyman Milanfar. The rate-distortion-accuracy tradeoff: JPEG case study. In *DCC*, pages 354–354, 2021. 3

[17] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Empirical Evaluation of Rectified Activations in Convolutional Network. In *ICML*, page 3, 2013. 2

[18] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 4

[19] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. 2

[20] Aoi Otani, Ryota Hashiguchi, Kazuki Omi, Norishige Fukushima, and Toru Tamaki. Performance evaluation of action recognition models on low quality videos. *IEEE Access*, 10:94898–94907, 2022. 5

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 3

[22] Kaitian Qiu, Lu Yu, and Daowen Li. Codec-simulation network for joint optimization of video coding with pre- and post-processing. *IEEE Open J. Circuits Syst.*, 2:648–659, 2021. 3

[23] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: An open source differentiable computer vision library for PyTorch. In *WACV*, pages 3674–3683, 2020. 3

[24] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419, 2020. 3, 4

[25] Yuan Tian, Guo Lu, Xiongkuo Min, Zhaohui Che, Guangtao Zhai, Guodong Guo, and Zhiyong Gao. Self-conditioned probabilistic learning of video rescaling. In *ICCV*, pages 4490–4499, 2021. 3

[26] Suramya Tomar. Converting video formats with FFmpeg. *Linux J.*, 2006(146):10, 2006. 3

[27] TorchVision Maintainers and Contributors. TorchVision: PyTorch's computer vision library. https://github.com/pytorch/vision, 2016. 3, 4

[28] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Trans. Consum. Electron.*, 38(1):xviii–xxxiv, 1992. 3, 5

[29] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *CVPR*, pages 606–615, 2018. 2

[30] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):560–576, 2003. 6

[31] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, pages 3–19, 2018. 2

[32] Xiufeng Xie, Ning Zhou, Wentao Zhu, and Ji Liu. Bandwidth-aware adaptive codec for dnn inference offloading in iot. In *ECCV*, pages 88–104, 2022. 3

[33] Lijun Zhao, Huihui Bai, Anhong Wang, and Yao Zhao. Learning a virtual codec based on deep convolutional neural network to compress image. *J. Vis. Commun. Image Represent.*, 63:102589, 2019. 3