A. Comparison with State-of-the-art

We give an overview of relevant recent learned compression methods and their differences to our GC method and BPG in Table 1. [34] were state-of-the-art in MS-SSIM in 2017, while work [31] is the current state-of-the-art in image compression in terms of classical metrics (PSNR and MS-SSIM) when measured on the Kodak data set [24]. Notably, all methods except ours (BPG, Rippel et al., and Minnen et al.) employ adaptive arithmetic coding using context models for improved compression performance. Such models could also be implemented for our system, and have led to additional savings of 10% in [30]. Since Rippel et al. and Minnen et al. have only released a selection of their decoded images (for 3 and 4, respectively, out of the 24 Kodak images), and at significantly higher bitrates, a comparison with a user study is not meaningful. Instead, we try to qualitatively put our results into context with theirs.

In Figs. 13–15 in Sec. F.4, we compare qualitatively to [34]. We can observe that even though Rippel *et al*. [34] use 29–179% more bits, our models produce images of comparable or better quality.

In Figs. 16–19 in Sec. F.5, we show a qualitative comparison of our results to the images provided by the work of [31], as well as to BPG [7] on those images. First, we see that BPG is still visually competitive with the current state-of-the-art, which is consistent with moderate 8.41% bitrate savings being reported by [31] in terms of PSNR. Second, even though we use much fewer bits compared to the example images available from [31], for some of them (Figs. 16 and 17) our method can still produce images of comparable visual quality.

B. Training Details

We employ the ADAM optimizer [23] with a learning rate of 0.0002 and set the mini-batch size to 1. Our networks are trained for 150000 iterations on Cityscapes and for 280000 iterations on Open Images. For normalization we used instance normalization [43], except in the second half of the Open Images training, we train the generator/decoder with fixed batch statistics (as implemented in the test mode of batch normalization [19]), since we found this reduced artifacts and color shift.

C. Data set and Preprocessing Details

To train GC models (which do not require semantic label maps, neither during training nor for deployment) for compression of diverse natural images, we use 200k images sampled randomly from the *Open Images* data set [25] (9M images). The training images are rescaled so that the longer side has length 768px, and images for which rescaling does not result in at least $1.25\times$ downscaling as well as high saturation images (average S > 0.9 or V > 0.8 in HSV color space) are discarded (resulting in an effective training set size of 188k).

We evaluate these models on the Kodak image compression data set [24] (24 images, 768×512px), which has a long tradition in the image compression literature and is still the most frequently used data set for comparisons of learned image compression methods. Additionally, we evaluate our GC models on 20 randomly selected images from the *RAISE1K* data set [11], a real-world image data set consisting of 8156 high-resolution RAW images (we rescale the images such that the longer side has length 768px). To investigate the benefits of having a somewhat constrained application domain and semantic labels at training time, we also train GC models with semantic label maps on the *Cityscapes* data set [9] (2975 training and 500 validation images, 34 classes, 2048×1024px resolution) consisting of street scene images and evaluate it on 20 randomly selected validation images (without semantic labels). Both training and validation images are rescaled to 1024×512px resolution.

To evaluate the proposed SC method (which requires semantic label maps for training and deployment) we again rely on the Cityscapes data set. Cityscapes was previously used to generate images form semantic label maps using GANs [20, 50]. The preprocessing for SC is the same as for GC.

D. Compression Details

We compress the semantic label map for SC by quantizing the coordinates in the vector graphic to the image grid and encoding coordinates relative to preceding coordinates when traversing object boundaries (rather than relative to the image frame). The so-obtained bitstream is then compressed using arithmetic coding.

To ensure fair comparison, we do not count header sizes for any of the baseline methods throughout.

E. Architecture Details

For the GC, the encoder E convolutionally processes the image $\boldsymbol{x}$ and optionally the label map $\boldsymbol{s}$, with spatial dimension $W \times H$, into a feature map of size ${}^{W}\!/_{16} \times {}^{H}\!/_{16} \times 960$ (with 6 layers, of which four have 2-strided convolutions), which is then projected down to C channels (where $C \in \{2, 4, 8\}$ is much smaller than 960). This results in a feature map $\boldsymbol{w}$ of dimension ${}^{W}\!/_{16} \times {}^{H}\!/_{16} \times C$, which is quantized over L centers to obtain the discrete $\hat{\boldsymbol{w}}$. The generator G projects $\hat{\boldsymbol{w}}$ up to 960 channels, processes these with 9 residual units [18] at dimension ${}^{W}\!/_{16} \times {}^{H}\!/_{16} \times 960$, and then mirrors E by convolutionally processing the features back to spatial dimensions $W \times H$ (with transposed convolutions instead of strided ones).

Similar to E, the feature extractor F for SC processes the semantic map $\boldsymbol{s}$ down to the spatial dimension of $\hat{\boldsymbol{w}}$,

	BPG	Rippel et al. (2017)	Minnen et al. (2018)	Ours (GC)
Learned	No	Yes	Yes	Yes
Arithmetic encoding	Adaptive	Adaptive	Adaptive	Static
Context model	CABAC	Autoregressive	Autoregressive	None
Visualized bitrates [bpp][4]	all[5]	0.08–	0.12–	0.033–0.066
GAN	No	Non-standard	No	f-div. based
S.o.t.a. in MS-SSIM	No	No	Yes	No
S.o.t.a. in PSNR	No	No	Yes	No
Savings to BPG in PSNR			8.41%	
Savings to BPG in User Study				17.2–48.7%

Table 1. Overview of differences between [31] (s.o.t.a. in MS-SSIM and PSNR), to BPG (previous s.o.t.a. in PSNR) and [34] (s.o.t.a. in MS-SSIM in 2017, also used GANs).

which is then concatenated to $\hat{\boldsymbol{w}}$ for generation. In this case, we consider slightly higher bitrates and downscale by $8\times$ instead of $16\times$ in the encoder E, such that $\dim(\hat{\boldsymbol{w}}) = {}^{W}/_{8} \times {}^{H}/_{8} \times C$. The generator then first processes $\hat{\boldsymbol{w}}$ down to ${}^{W}/_{16} \times {}^{H}/_{16} \times 960$ and then proceeds as for GC.

For both GC and SC, we use the multi-scale architecture of [44] for the discriminator D, which measures the divergence between $p_{\boldsymbol{x}}$ and $p_{G(\boldsymbol{z})}$ both locally and globally.

We adopt the notation from [44] to describe our encoder and generator/decoder architectures and additionally use `q` to denote the quantization layer (see Sec. 3 for details). The output of `q` is encoded and stored.

- **Encoder GC:** `c7s1-60, d120, d240, d480, d960, c3s1-`C`, q`
- **Encoders SC:**
  - Semantic label map encoder: `c7s1-60, d120, d240, d480, d960`
  - Image encoder: `c7s1-60, d120, d240, d480, c3s1-`C`, q, c3s1-480, d960`

  The outputs of the semantic label map encoder and the image encoder are concatenated and fed to the generator/decoder.
- **Generator/decoder:** `c3s1-960, R960, R960, R960, R960, R960, R960, R960, R960, u480, u240, u120, u60, c7s1-3`

F. Visuals

In the following Sections, F.1, F.2, F.3, we show the first five images of each of the three data sets we used for the user study, next to the outputs of BPG at similar bitrates.

Secs. F.4 and F.5 provide visual comparisons of our GC models with [34] and [31], respectively, on a subset of images form the Kodak data set.

In Sec. F.6, we show visualizations of the latent representation of our GC models.

Finally, Sec. F.7 presents additional visual results for SC.

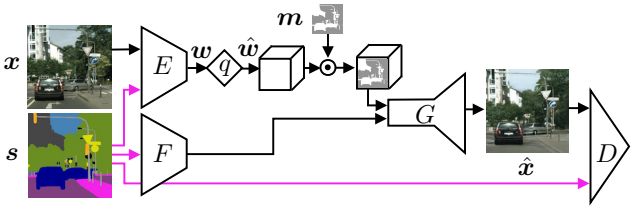


Figure 9. Structure of the proposed SC network. E is the encoder for the image $\boldsymbol{x}$ and the semantic label map $\boldsymbol{s}$. q quantizes the latent code $\boldsymbol{w}$ to $\hat{\boldsymbol{w}}$. The subsampled heatmap multiplies $\hat{\boldsymbol{w}}$ (pointwise) for spatial bit allocation. G is the generator/decoder, producing the decompressed image $\hat{\boldsymbol{x}}$, and D is the discriminator used for adversarial training. F extracts features from $\boldsymbol{s}$.

F.1. Generative Compression on Kodak

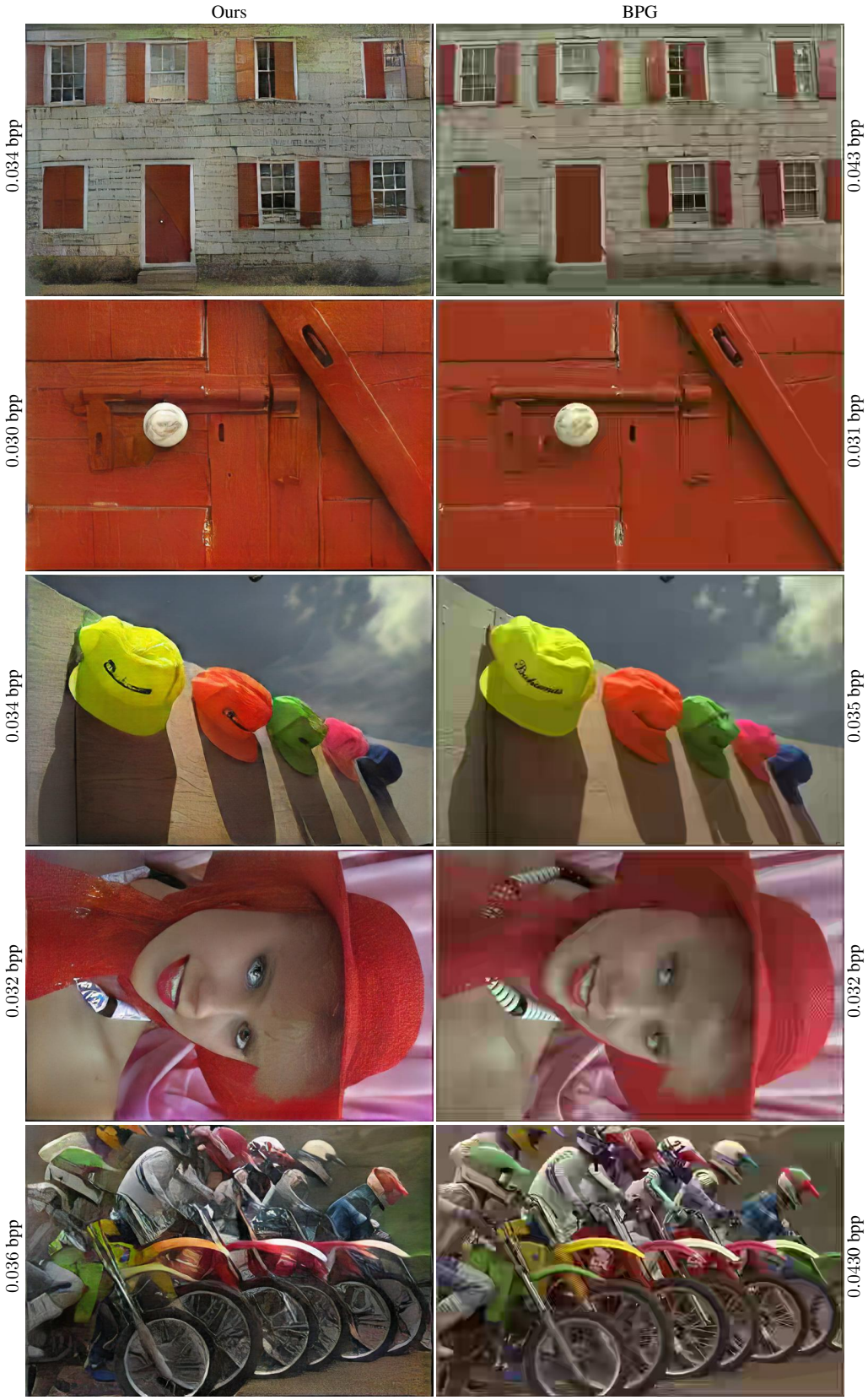


Figure 10. First 5 images of the Kodak data set, produced by our GC model with $C = 4$ and BPG.

F.2. Generative Compression on RAISE1k

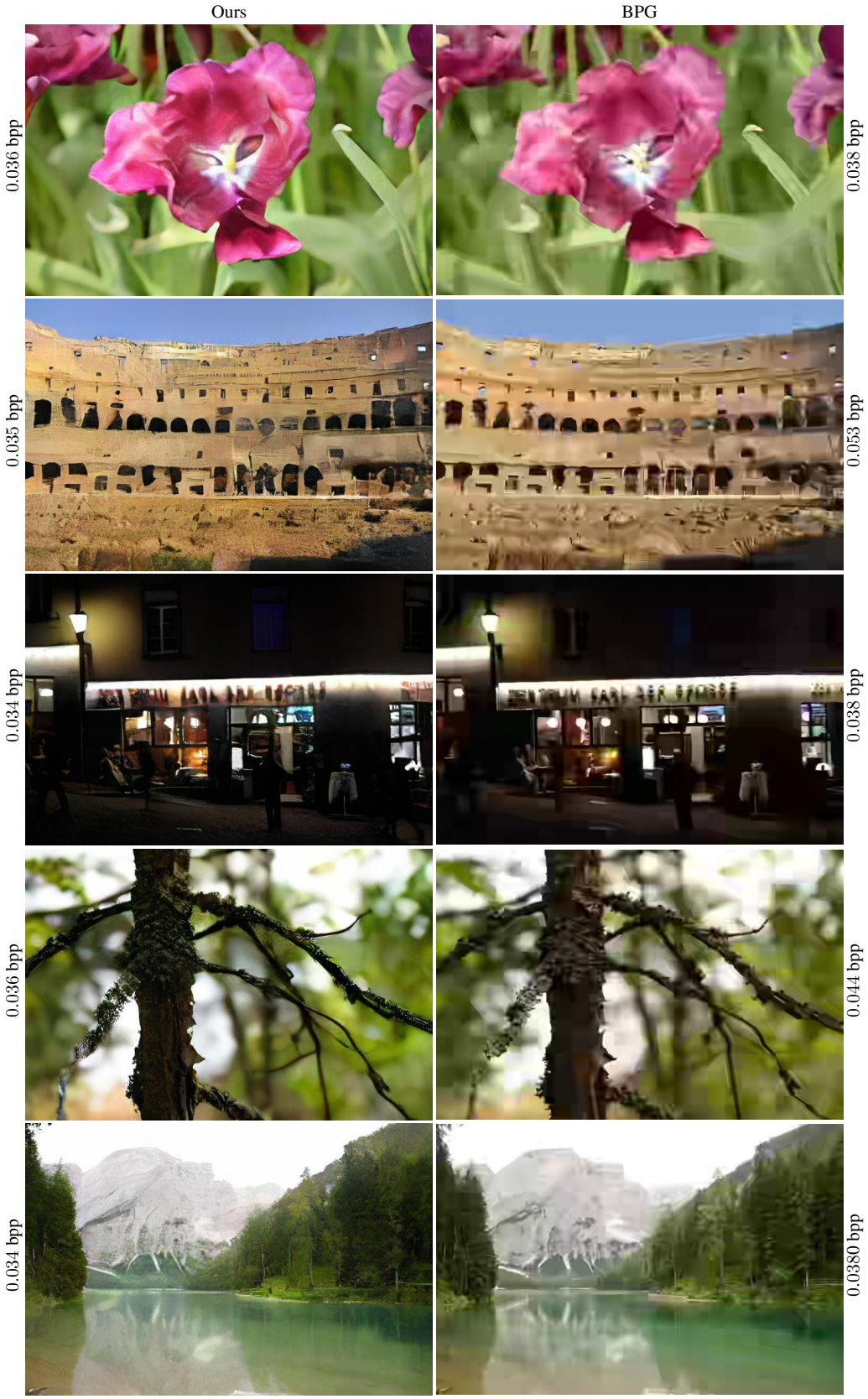


Figure 11. First 5 images of RAISE1k, produced by our GC model with $C = 4$ and BPG.

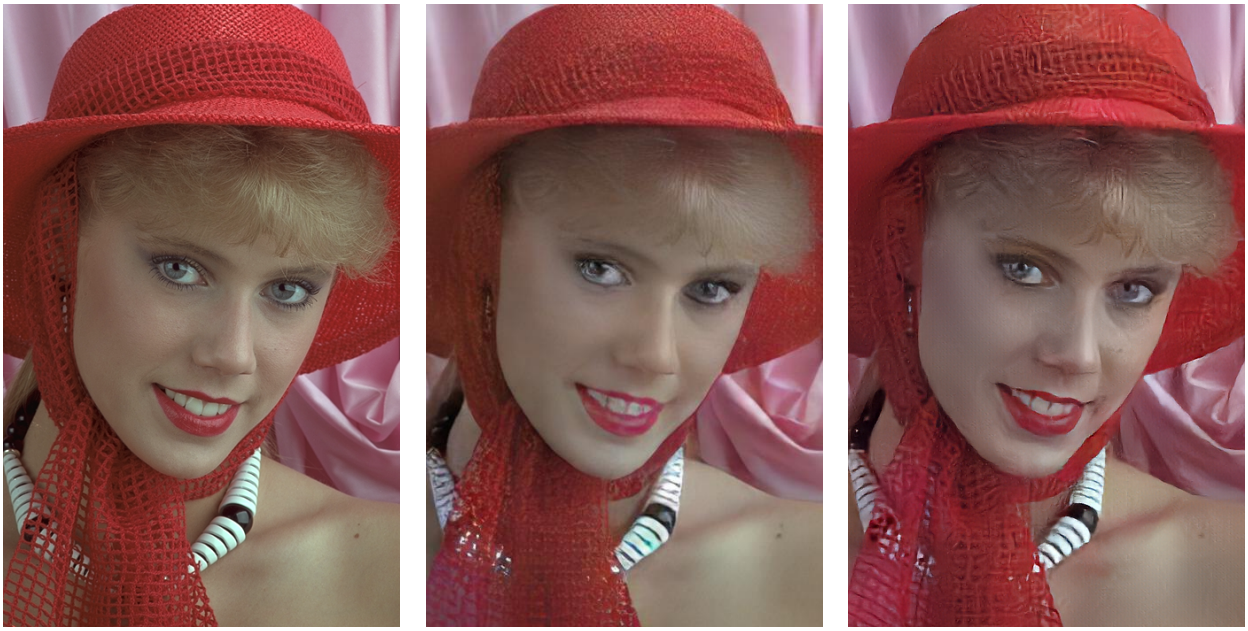
F.3. Generative Compression on Cityscapes

Figure 12. First 5 images of Cityscapes, produced by our GC model with $C = 4$ and BPG.

F.4. Comparison with [34]

Original | Ours, 0.0304bpp | Rippel et al., 0.0828bpp (+*172%*)

Figure 13. Our model loses more texture but has less artifacts on the knob. Overall, it looks comparable to the output of [34], using significantly fewer bits.



Original | Ours, 0.0651bpp | Rippel et al., 0.0840bpp (+*29%*)

Figure 14. Notice that compared to [34], our model produces smoother lines at the jaw and a smoother hat, but provides a worse reconstruction of the eye.

Original | Ours, 0.0668bpp | Rippel et al., 0.0928bpp (+*39%*)

Figure 15. Notice that our model produces much better sky and grass textures than [34], and also preserves the texture of the light tower more faithfully.

F.5. Comparison with [31]

Original

Ours, 0.0668bpp

Minnen et al., 0.221bpp *230% larger*

BPG, 0.227bpp

Figure 16. Notice that our model yields sharper grass and sky, but a worse reconstruction of the fence and the lighthouse compared to [31]. Compared to BPG, Minnen et al. produces blurrier grass, sky and lighthouse but BPG suffers from ringing artifacts on the roof of the second building and the top of the lighthouse.

Original

Ours, 0.0685bpp

Minnen et al., 0.155bpp, *127% larger*

BPG, 0.164bpp

Figure 17. Our model produces an overall sharper face compared to [31], but the texture on the cloth deviates more from the original. Compared to BPG, Minnen et al. has a less blurry face and fewer artifacts on the cheek.

Original

Ours, 0.0328bpp

Minnen et al., 0.246bpp, *651% larger*

BPG, 0.248bpp

Figure 18. Here we obtain a significantly worse reconstruction than [31] and BPG, but use only a fraction of the bits. Between BPG and Minnen et al., it is hard to see any differences.

Original

Ours, 0.03418bpp

Minnen et al., 0.123bpp, *259% larger*,

BPG, 0.119bpp

Figure 19. Here we obtain a significantly worse reconstruction compared to [31] and BPG, but use only a fraction of the bits. Compared to BPG, Minnen et al.has a smoother background but less texture on the birds.

F.6. Sampling the compressed representations

Figure 20. We uniformly sample codes from the (discrete) latent space $\hat{\boldsymbol{w}}$ of our generative compression models (GC with $C = 4$) trained on Cityscapes and Open Images. The Cityscapes model outputs domain specific patches (street signs, buildings, trees, road), whereas the Open Images samples are more colorful and consist of more generic visual patches.

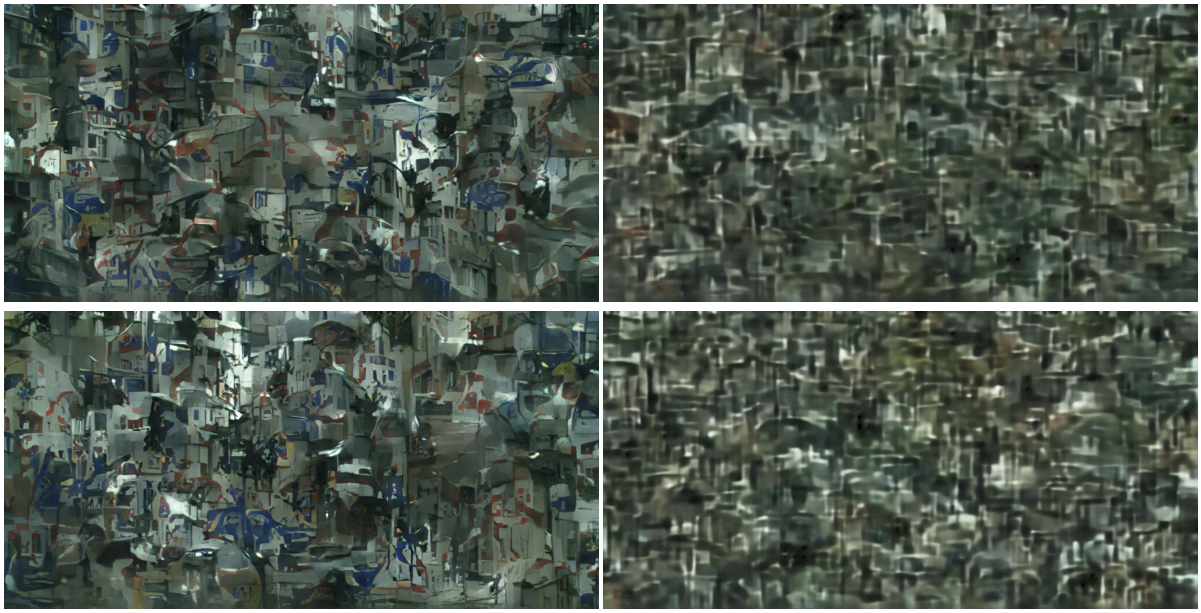


Figure 21. We train the same architecture with $C = 4$ for MSE and for generative compression on Cityscapes. When uniformly sampling the (discrete) latent space $\hat{\boldsymbol{w}}$ of the models, we see stark differences between the decoded images $G(\hat{\boldsymbol{w}})$. The GC model produces patches that resemble parts of Cityscapes images (street signs, buildings, etc.), whereas the MSE model outputs looks like low-frequency noise.

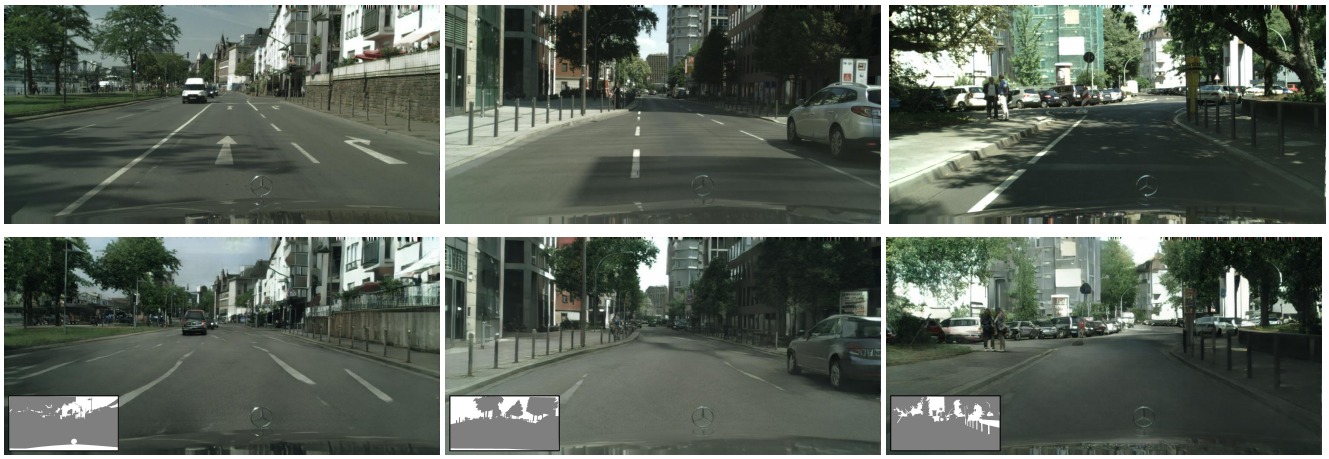
GC model with $C = 4$ MSE baseline model with $C = 4$

Figure 22. We experiment with learning the distribution of $\hat{\boldsymbol{w}} = E(\boldsymbol{x})$ by training an improved Wasserstein GAN [14]. When sampling form the decoder/generator G of our model by feeding it with samples from the improved WGAN generator, we obtain much sharper images than when we do the same with an MSE model.

F.7. Selective Compression on Cityscapes

Figure 23. Synthesizing different classes for two different images from Cityscapes, using our SC network with $C = 4$. In each image except for *no synthesis*, we additionally synthesize the classes *vegetation, sky, sidewalk, ego vehicle, wall*.

Figure 24. Example images obtained by our SC network ($C = 8$) preserving a box and synthesizing the rest of the image, on Cityscapes. The SC network seamlessly merges preserved and generated image content even in places where the box crosses object boundaries.



0.019 bpp 0.021 bpp 0.013 bpp

Figure 25. Reconstructions obtained by our SC network using semantic label maps estimated from the input image via PSPNet [49].