A Supplementary Material

A.1 OASIS-C Architecture

We summarize our discriminator architecture in Tab. 3. Our architecture is split into two parts. The first part is identical to OASIS (Schönfeld *et al.*, 2021), except that we replace spectral norm with weight norm. The output `out` is a $256 \times 256 \times N + 1$ prediction map. In the second part, we adopt a pixel-wise projection-based conditioning scheme. We use a similar latent pre-processing block as in HiFiC, but use 64 instead of 12 filters. The pre-processed latent feature map `y_prep` has an identical shape as `out` and is subsequently incorporated into the discriminator, using projection (element-wise multiplication and sum across the channel dimension). The projected feature map `proj` is finally replicated and added back to `out`.

Table 3: OASIS-C Architecture

Operation	Input	Size	Output	Size
ResBlock-Down	`image` (x)	$256 \times 256 \times 3$	`d1`	$128 \times 128 \times 128$
ResBlock-Down	`d1`	$128 \times 128 \times 128$	`d2`	$64 \times 64 \times 128$
ResBlock-Down	`d2`	$64 \times 64 \times 128$	`d3`	$32 \times 32 \times 256$
ResBlock-Down	`d3`	$32 \times 32 \times 256$	`d4`	$16 \times 16 \times 256$
ResBlock-Down	`d4`	$16 \times 16 \times 256$	`d5`	$8 \times 8 \times 512$
ResBlock-Down	`d5`	$8 \times 8 \times 512$	`d6`	$4 \times 4 \times 512$
ResBlock-Up	`d6`	$4 \times 4 \times 512$	`u1`	$8 \times 8 \times 512$
ResBlock-Up	`cat(u1, d5)`	$8 \times 8 \times 1024$	`u2`	$16 \times 16 \times 256$
ResBlock-Up	`cat(u2, d4)`	$16 \times 16 \times 512$	`u3`	$32 \times 32 \times 256$
ResBlock-Up	`cat(u3, d3)`	$32 \times 32 \times 512$	`u4`	$64 \times 64 \times 128$
ResBlock-Up	`cat(u4, d2)`	$64 \times 64 \times 256$	`u5`	$128 \times 128 \times 128$
ResBlock-Up	`cat(u5, d1)`	$128 \times 128 \times 256$	`u6`	$256 \times 256 \times 64$
Conv2D	`u6`	$256 \times 256 \times 64$	`out`	$256 \times 256 \times N + 1$
Conv2D, Resize	`latent` (y)	$16 \times 16 \times C_y$	`y_prep`	$256 \times 256 \times 64$
Projection	`(u6, y_prep)`	$256 \times 256 \times 64$	`proj`	$256 \times 256 \times 1$
Add	`(out, proj)`	$256 \times 256 \times N + 1$	D_{out}	$256 \times 256 \times N + 1$

A.2 Pre-trained Semantic Segmentation Performance

We pre-train the first discriminator block (Fig. 2, highlighted orange) using DeepLab2[7]. The resulting semantic segmentation performance measured by the mean intersection over union (mIoU) is summarized in Tab. 4.

[7] We base our experiments on the panoptic configurations presented in `https://github.com/google-research/deeplab2/blob/main/g3doc/projects/panoptic_deeplab.md` (ResNet-50).

Table 4: Pre-trained Semantic Segmentation Performance

Dataset	Crop Size	Batch Size	Steps	mIoU ↑
Cityscapes (Cordts *et al.*, 2016)	256×256	16	320k	0.67
Coco2017 (Lin *et al.*, 2014)	256×256	16	1M	0.41

A.3 Additional Experimental Details

Preliminary study. We use the official implementation for PatchGAN and translate the SESAME, U-Net, projected and OASIS discriminators carefully to TensorFlow, based on the official PyTorch implementations. For projected GANs, we use the efficientnet-lite4-variant (Tan *et al.*, 2019) as a pre-trained feature network, which we have found to produce the best results.

To maintain the advantages of having pre-trained feature extractors, we have used a slightly different concatenation-based conditioning scheme for conf-d and conf-f; for conf-d, we pre-process and concatenate the latent features with the efficientnet-lite4-based feature maps at each scale separately. For conf-f, we integrate y using a similar HiFiC-based concatenation scheme (see Fig. 12).

For conf-c (projection), we use two separate latent pre-processing blocks with channel dimensions 64 and 4, corresponding to the local and global outputs, respectively. We use no resize operation for the latter to match the feature dimension prior to classification ($16 * 16 * 4 = 1024$).

Main study. We train six models for 2+1M optimization steps, using $\lambda \in \{2, 1.5, 1, 0.5, 0.25, 0.1\}$, a crop size of 256 and a batch size of 16 and 8 for stage one and two, respectively. We use the Adam optimizer with default settings ($\beta_1 = 0.9, \beta_2 = 0.999$). For stage one, we use a learning rate of 1e−4 for the first 1.8M steps and subsequently decay the learning rate to 1e−5, similar to previous work. For stage two, we use the same settings as in Ours w/ d (HiFiC), *i.e.*, training strategy-I with a fixed learning rate of 1e−5, except for β (Eq. (4)), which we increase from 0.15 to 0.30.

ORP. We finetune G_{ORP} for additional 2M steps. In practice we have found it slightly more efficient to directly predict the MSE-optimized decoder output MSE_{pred} and to calculate $R = MSE_{pred} - G_2(x)$.

Note that ORP is a general formulation for multi-realim image compression, which allows for different model parameterizations. By increasing the model capacity of G_{ORP} up to G_2, we can approach the performance of traditional image/ weight interpolation techniques (see Appendix A.13).

A.4 Comparing Normalization Strategies

We summarize some of the normalization methods we tried for OASIS in Tab. 5 and Fig. 16. For spectral normalization, we have found that tuning the Lipschitz constant is indeed helpful. However, we did not find a configuration that

Table 5: The effect of different normalization strategies on the semantic segmentation and its resulting compression performance.

Method	mIoU ↑	PSNR ↑	FID ↓	Batch Size
Spectral norm (Miyato *et al.*, 2018)	0.49	30.01	9.75	16
Weight norm (Salimans *et al.*, 2016)	0.67	**30.20**	**7.96**	16
Layer norm (Ba *et al.*, 2016)	**0.68**	29.49	9.00	8

exceeded the performance of weight normalization and hence omit it here. For layer normalization we had to reduce the batch size to 8, due to out-of-memory issues.

A.5 LabelMix Regularization

As mentioned in the main paper (Eq. (6)), we regularize the discriminator with the LabelMix consistency loss (Schönfeld *et al.*, 2021), tailored to the compression setting. We provide additional intuition in Figs. 17 and 18.

A.6 Performance on DIV2K

In Fig. 7 we provide an extended comparison to the state-of-the-art on DIV2K. We observe similar trends as discussed for CLIC 2020, except that our method outperforms MS-ILLM in terms of FID in the low bit range.

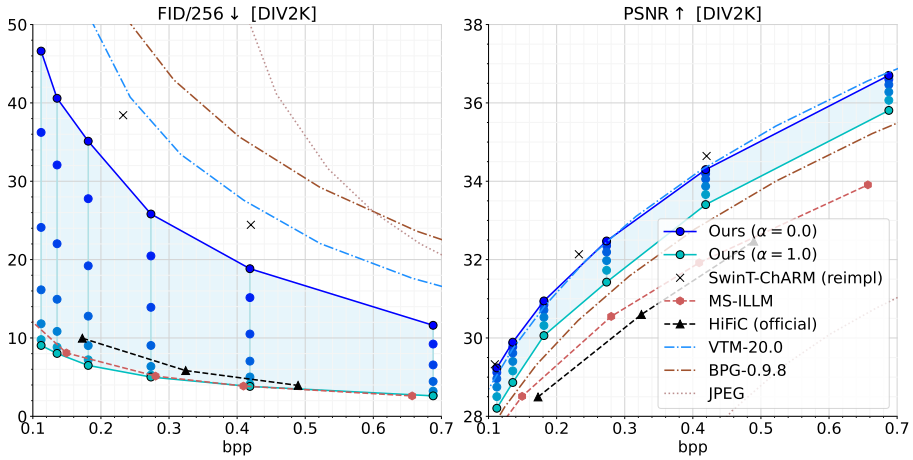

Fig. 7: Comparison to the state-of-the-art on DIV2K

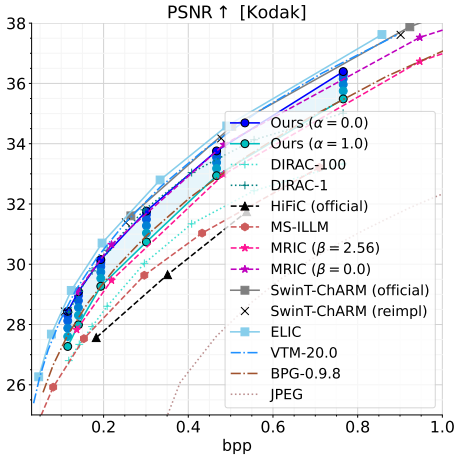

Fig. 8: Kodak rate-distortion plot

A.7 Performance on Kodak

In Fig. 8 we provide the rate-distortion performance for the Kodak dataset. We add the official values of SwinT-ChARM (Zhu *et al.*, 2022) and ELIC (He *et al.*, 2022) for reference. Exact configurations for JPEG, BPG-0.9.8, and VTM-20.0 can be found in Appendices A.16 to A.18.

Note that SwinT-ChARM is almost on par with the current state-of-the-art method ELIC in terms of PSNR and thus represents a good base model for our work. The marginal gap is due to ELIC's more powerful entropy model. We emphasize that both EGIC, MRIC, and DIRAC rely on some variant of the ChARM-entropy model (Minnen *et al.*, 2020).

Similar to MRIC, we observe that introducing higher perception results in a $1 - 1.5$dB PSNR decrease.

A.8 SwinT-ChARM Reimplementation

In Fig. 8, we compare the compression performance of our SwinT-ChARM reimplementation (reimpl) to the official values, measured on the Kodak dataset. We optimized the reimplemented version for 2M optimization steps on the CLIC 2020 training set, using $\lambda \in \{0.01, 0.003, 0.001, 0.0003\}$ and a batch size of 8. We used a learning rate of 1e$-$4 for the first 1.8M steps and subsequently decayed the learning rate to 1e$-$5. We find that our reimplementation closely matches the official values (up to 0.1dB tolerance), despite being trained from scratch and using less than two-thirds of the optimization steps.

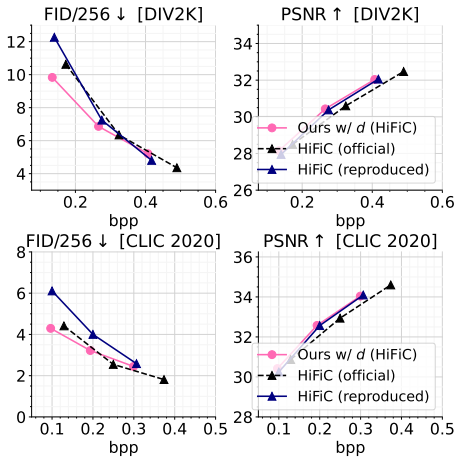

Fig. 9: Relative comparison to HiFiC

A.9 Experiments on HiFiC

In Fig. 9, we compare Ours w/ d (HiFiC) to HiFiC (reproduced). Note that HiFiC (official) was trained on an internal dataset is therefore only visualized for transparency reasons. We observe that Ours w/ d (HiFiC) is most effective in the low to medium bit range, which is the key focus of our work. For HiFiC-lo, we achieve an improvement of up to 2 FID points with slightly better PSNR (+0.2dB), suggesting that our method is particularly well suited for the extremely low bit range < 0.1bpp.

For HiFiC-hi, we experience a slight decrease in performance. We suspect that this is due to misaligned hyper-parameters; indeed recent work suggests that a separate set of hyper-parameters is required for various bit-rates (Muckley *et al.*, 2023).

A.10 Comparing Training Dynamics

In Fig. 10, we compare the training dynamics of OASIS w/ d and Ours w/ d. We find that OASIS with weight norm greatly increases model capacity, while pre-training accelerates training, resulting in superior compression performance. Note that our method provides robust and stable training across different compression rates, while OASIS w/ d exhibits training instabilities that are particularly evident on complex datasets (Coco2017).

In Fig. 11, we provide further performance insights into the training dynamics of Ours w/ d (HiFiC) and HiFiC (reproduced) for stage two. We report the means and standard deviations of BPP, PSNR, and FID as a function of the number of optimization steps across two test runs. Note that HiFiC's training procedure is divided into 3 phases: warm-up ($0 - 50$k), training with a learning rate of 1e$-$4

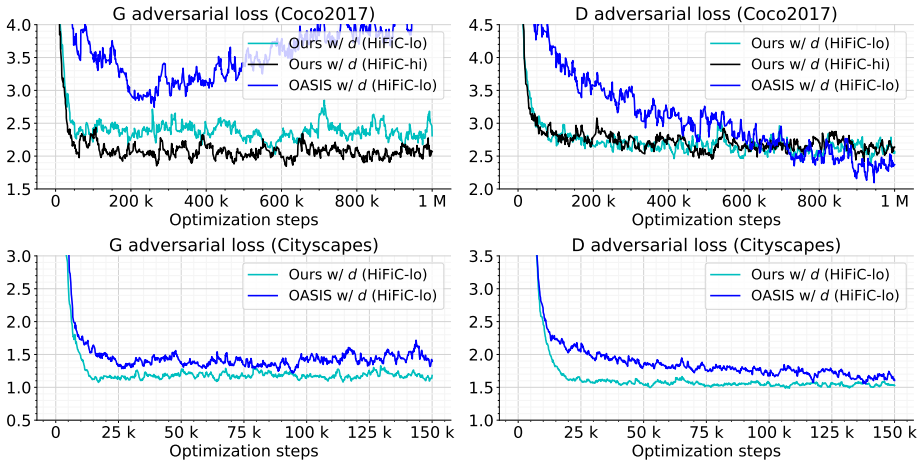

Fig. 10: Comparing the training dynamics of OASIS w/ d and Ours w/ d. "G adversarial" corresponds to the $(N+1)$-cross entropy loss and hence gives an idea of how realistic and semantically correct the resulting reconstructions are (lower is better). "D adversarial" includes regularization terms (lower is better).

$(50-500\text{k})$ and 1e−5 (500k-1M), respectively, whereas, Ours w/ d (HiFiC) uses the same learning rate and λ-schedule across all training steps.

We find that our method considerably accelerates training progress, similar to projected GANs (Sauer *et al.*, 2021). As can be seen, our method exceeds the performance of HiFiC after only 300k optimization steps. The large deviations at the beginning of the training phase can be attributed to a sort of calibration phase in which the variables for the projection-based conditioning mechanism are learned from scratch.

A.11 Impact of the Focal Frequency Loss

In Tab. 6, we summarize the effect of the focal frequency loss (FFL, Jiang *et al.*, 2021) on the concat base configurations. For that, we finetune all base models for additional 50k steps. We find that the FFL has the greatest impact on conf-a and conf-d, whereas it has little impact on the discriminators based on pixel-level supervision (conf-c and conf-e). We also find that the FFL cannot further improve ours w/o d, which reinforces the design decisions made in our work.

A.12 Computational/ Model Complexity

In Tab. 7, we compare the storage-efficiency of each model in terms of model parameters (in millions). For the generator, we further differentiate between the base model size and additional parameters required for traversing the D-P curve (denoted by +). The calculation for P includes the hyper-analysis and

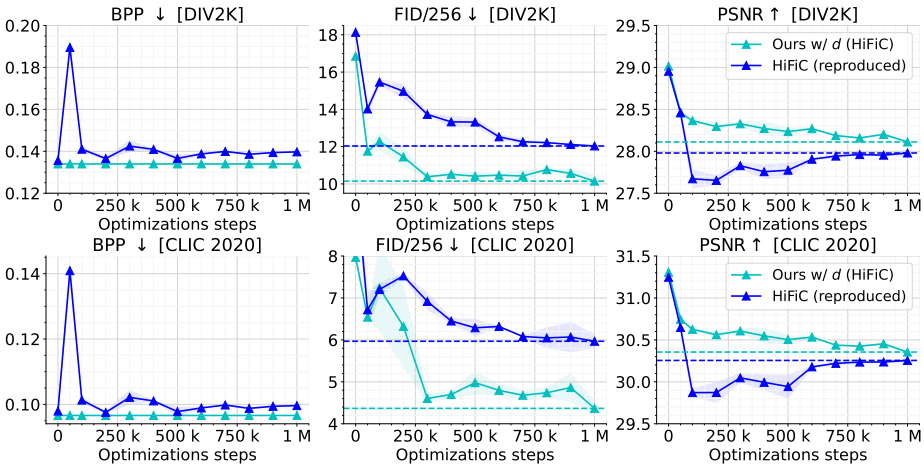

Fig. 11: BPP, FID, and PSNR vs optimization steps for the second stage of Ours w/ d (HiFiC) and HiFiC (reproduced). We show the mean and standard deviation across 2 runs per setting. Values at step 0 correspond to the output values from stage one. We additionally show the values at step 50k (warm-up phase in HiFiC).

hyper-synthesis transforms, as well as additional slice transforms in the case of ChARM.

It is worth noting that EGIC during inference is identical to SwinT-ChARM (neglecting ORP); latency numbers can be found in Zhu *et al.*, 2022 (Tab. 3 and Sec. D.3). GPU-memory overhead only incurs during training.

A.13 Image/ Weight Interpolation

Image and weight interpolation (Wang *et al.* 2019, Iwai *et al.* 2021, Yan *et al.* 2022) can be achieved using

$$x' = (1-\alpha)G_1(y) + \alpha G_2(y), \tag{9}$$

$$x' = G_\theta(y); \theta = (1-\alpha)\theta_{G_1} + \alpha\theta_{G_2}, \tag{10}$$

where θ and α correspond to the model parameters and interpolation weight, respectively. For our ablation study, we fine-tune the generator weights from stage one G_1 ($= G_{ORP}$) for additional 500k optimization steps. We use $\alpha \in \{0.0, 0.17, 0.33, 0.5, 0.67, 0.83, 1.0\}$, resulting in seven points per bit-rate.

Our results[8] are summarized in Fig. 13 and Fig. 14. As can be seen, both methods work reasonably well; for weight interpolation, we observe skewed interpolation characteristics in some cases (*e.g.*, CLIC 2020 at low bit-rate). Notewor-

[8] The results are based on an early stage of EGIC, which produces a slightly different D-P trade-off. The overall logic remains however the same.

Table 6: Which method benefits the most from the FFL? The relative change (rel-PSNR, rel-FID) is here denoted over their respective concat-base configurations.

Method	Distortion		Perception	
	PSNR ↑	rel-PSNR	FID ↓	rel-FID
conf-a w/ FFL	**32.53**	+33.8%	40.64	-63.8%
conf-b w/ FFL	29.17	-0.9%	79.55	+5.2%
conf-c w/ FFL	29.23	-0.8%	89.73	+3.1%
conf-d w/ FFL	29.89	+1.7%	21.85	-29.1%
conf-e w/ FFL	30.25	+0.7%	15.93	-3.5%
ours w/o d	29.56	-1.4%	**8.73**	+12.8%

Table 7: Model size comparison in millions of parameters (M)

Method	E	G	P	Total (M)
HiFiC (Mentzer *et al.*, 2020)	7.4	156.8	17.3	181.5
MS-ILLM (Muckley *et al.*, 2023)	7.4	156.8	17.3	181.5
DIRAC (Ghouse *et al.*, 2023)	7.0	7.0 + 108.4	14.3	136.8
HFD/DDPM (Hoogeboom *et al.*, 2023)	10.7	10.7 + 1033.9	36.4	1091.7
MRIC (Agustsson *et al.*, 2023)	10.7	10.7 + 2.65	36.4	60.45
EGIC (Ours)	9.1	9.1 + 0.4	14.4	33

thy, Ours | interpol ($\alpha = 0.0$) almost matches the performance of SwinT-ChARM (reimpl), which can be considered an upper bound.

A.14 Visual Comparison: Concat vs Projection

In Fig. 15, we provide additional visual impressions of the effect of various conditioning strategies. We find that projection greatly helps to reduce image artifacts.

A.15 Pixel Weighting Schemes

Pixel weighting schemes have played a minor role in our work. As mentioned earlier, we use the simple instance size-based weighting scheme introduced in Yang *et al.* (2019), whereas, in Schönfeld *et al.* (2021), each semantic class is weighted by its inverse per-pixel frequency, computed over a batch of images. In Tab. 8, we show that this method is indeed effective and performs comparably to the more sophisticated approach of Schönfeld *et al.* (2021). In Fig. 19, we provide additional visual comparisons.

Table 8: Comparing different pixel weighting schemes

Method	PSNR ↑	FID ↓
OASIS (instance size-oriented)	29.90	**15.30**
OASIS (class-oriented)	29.90	15.56

A.16 Comparison to VVC-intra

The evaluation of the VVC standard (current state-of-the-art for non-learned image compression codecs) is based on VTM-20.0, a reference software provided by `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/VTM-20.0`. Similar to previous work, we first convert the PNG images to YCbCr-format using ffmpeg `https://www.ffmpeg.org/`:

```
ffmpeg
-i $PNGPATH -pix_fmt yuv444p $YUVPATH
```

To compress/ decompress the images, we use:

```
# Encode
EncoderAppStatic
-c encoder_intra_vtm.cfg -i $YUVPATH -q $Q,
-o /dev/null -b $OUTPUT
--SourceWidth=$WIDTH
--SourceHeight=$HEIGHT
--FrameRate=1 --FramesToBeEncoded=1
--InputBitDepth=8
--InputChromaFormat=444
--ConformanceWindowMode=1

# Decode
DecoderAppStatic
-b $OUTPUT -o $RECON -d 8
```

To convert the outputs back to PNG-format, we use:

```
ffmpeg
-f rawvideo -s $WIDTHx$HEIGHT
-pix_fmt yuv444p -i $RECON $RECON_PNG
```

PSNR is measured on the 8bit-decoded images and not on the floating point reconstructions, which is consistent with all our comparisons.

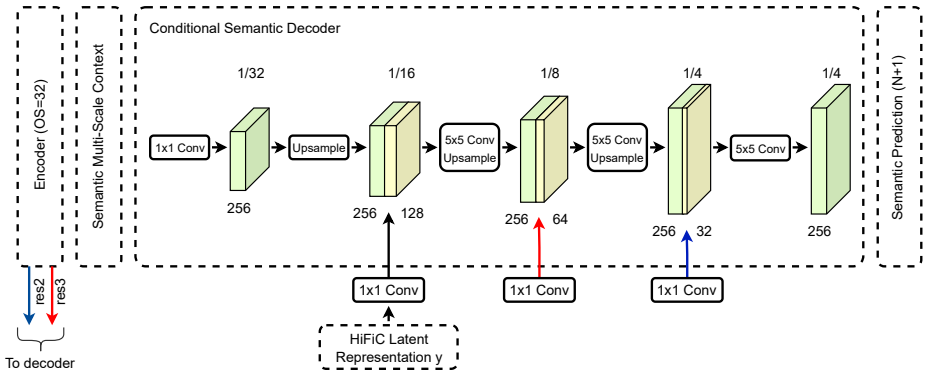

Fig. 12: Conditional panoptic DeepLab-based semantic decoder (conf-f)

A.17 Comparison to BPG

The evaluation of BPG-0.9.8 is based on the HEVC open video compression standard, provided by `https://bellard.org/bpg/`:

```
# Encode
bpgenc -o $OUTPUT -q $Q
-f 444 -e x265 -b 8 $INPUT

# Decode
bpgdec -o $RECON $OUTPUT
```

A.18 Comparison to JPEG

We use the Python Imaging Library (PIL) to obtain the JPEG encoded/ decoded images; (chroma) subsampling is set to 0, which corresponds to $4:4:4$, the highest quality setting.

```
tmp = io.BytesIO()
img.save(tmp, format='jpeg',
         subsampling=0,
         quality=Q)
tmp.seek(0)
filesize = tmp.getbuffer().nbytes
bpp = filesize * float(8)/
      img.size[0] * img.size[1]
rec = Image.open(tmp)
```

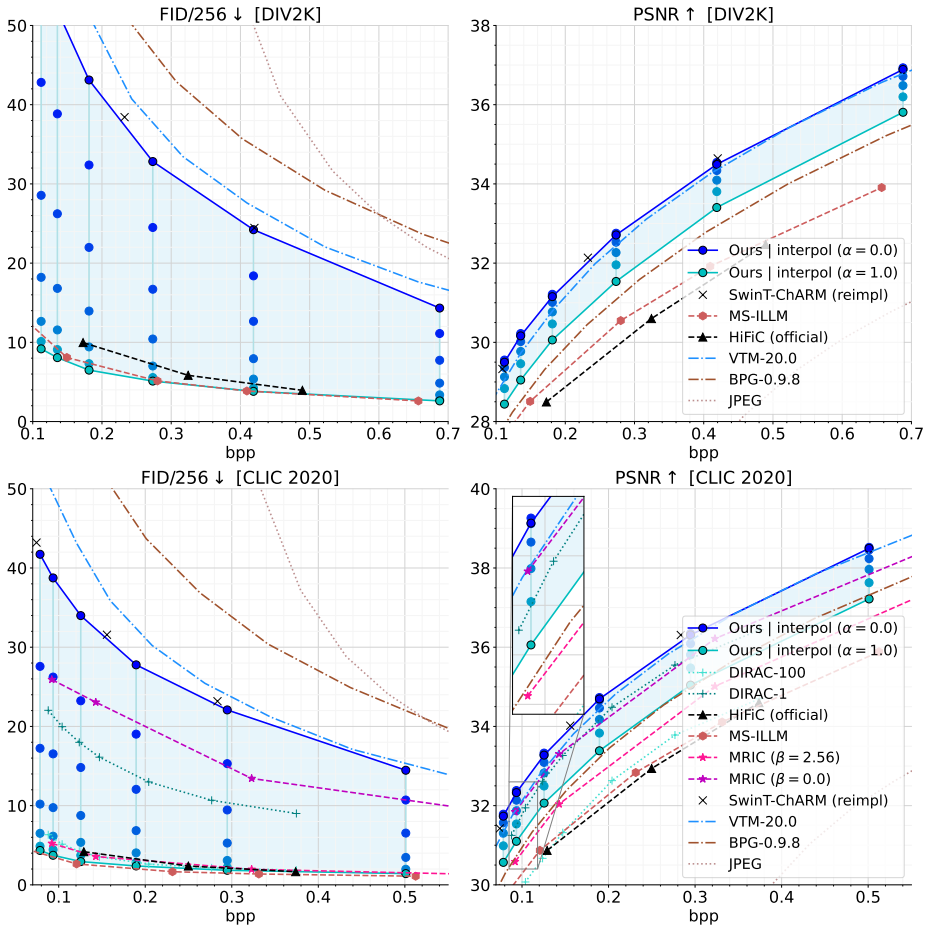

Fig. 13: Traversing the rate-distortion-perception plane using image interpolation

A.19 Visual Comparison

We provide extensive visual comparison to JPEG, BPG-0.9.8 and VTM.20.0 in Figs. 20 and 21, to HiFiC and MS-ILLM in Figs. 22 to 25, to MRIC ($\beta = 2.56$) and DIRAC-100 in Figs. 26 and 27, to PO-ELIC in Fig. 28 and to HFD/DDPM in Figs. 29 and 30.

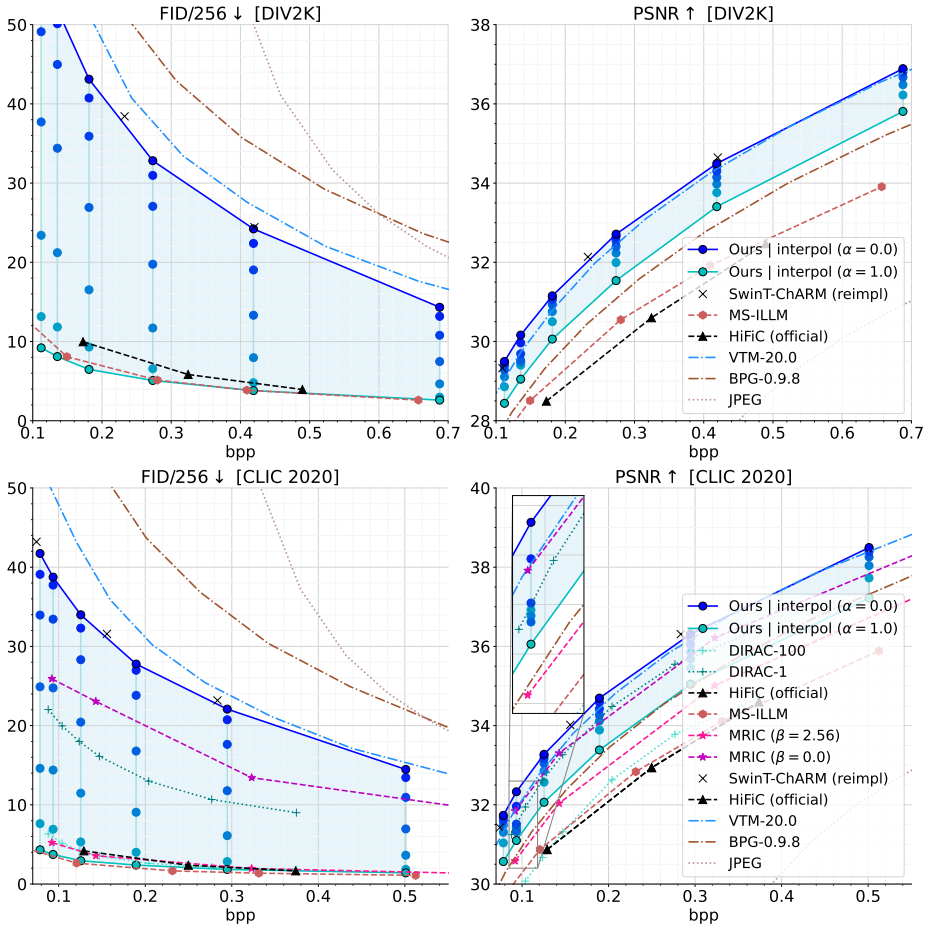

Fig. 14: Traversing the rate-distortion-perception plane using weight interpolation

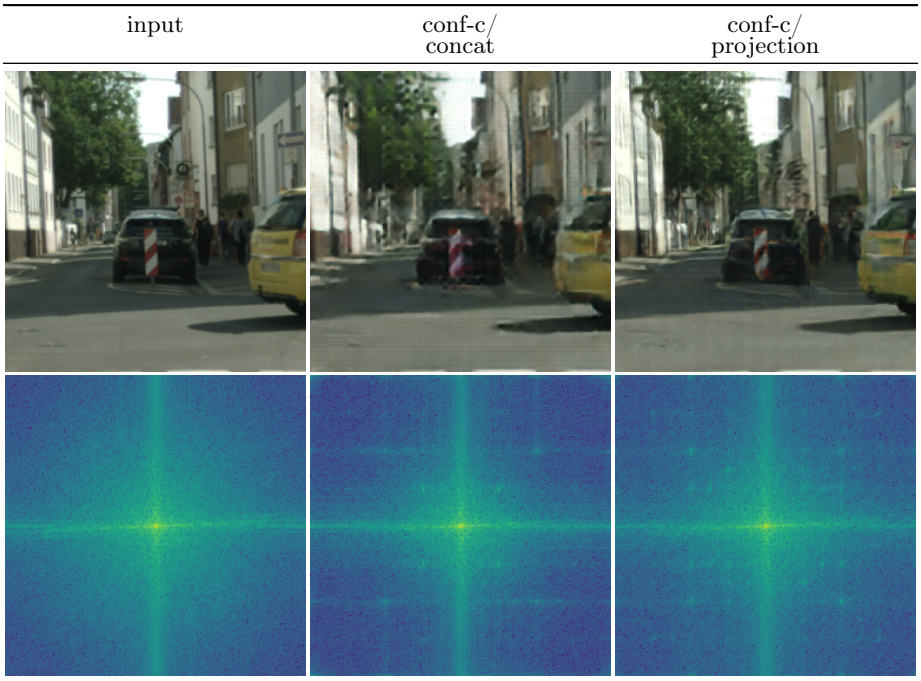

Fig. 15: Comparing U-Net (conf-c w/ concat) vs U-Net (conf-c w/ projection). Note that projection considerably reduces image artifacts.

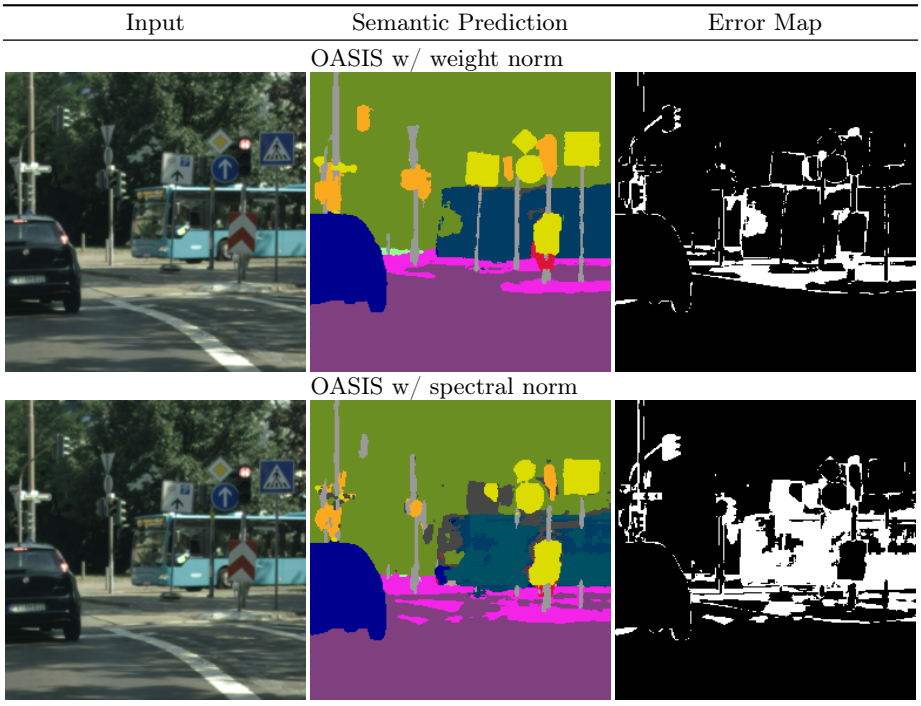

Fig. 16: Comparing the semantic segmentation performance of OASIS w/ weight normalization and OASIS w/ spectral normalization. Black pixels in the error map correspond to perfect prediction, white pixels highlight deviations from the ground truth.

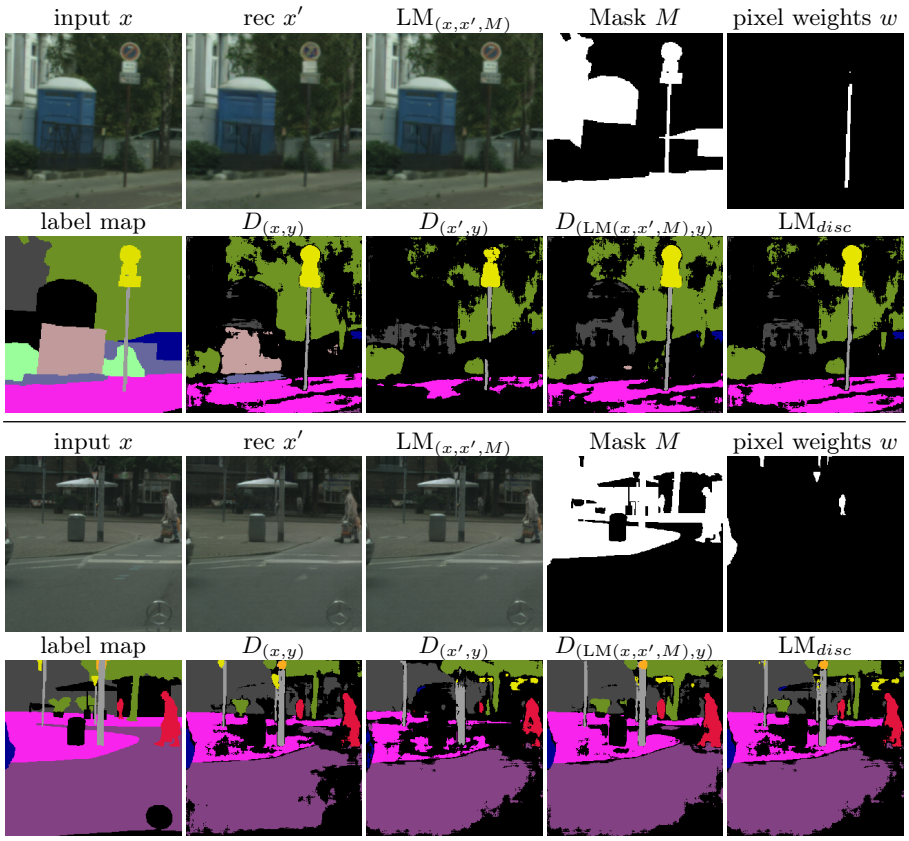

Fig. 17: Visualizing the discriminator loss components at step 33.5k (top row) and 35.2k (bottom row) on the Cityscapes dataset. We use an abbreviated notation in some cases due to space constraints; $D_{(\mathrm{LM}(x,x',M),y)}$ and LM_{disc} correspond to $D_{\mathrm{logits}}(\mathrm{LM}(x,x',M),y)$ and $\mathrm{LM}(D_{\mathrm{logits}}(x,y),D_{\mathrm{logits}}(x',y),M)$ in Eq. (6). We additionally visualize the pixel weight masks w introduced in Sec. 4 that highlight small object instances, as well as the corresponding label maps and the discriminator predictions for (x,y) and (x',y), respectively. The colorized discriminator predictions are obtained by $\arg\max(D)$. The black color corresponds to the fake class.

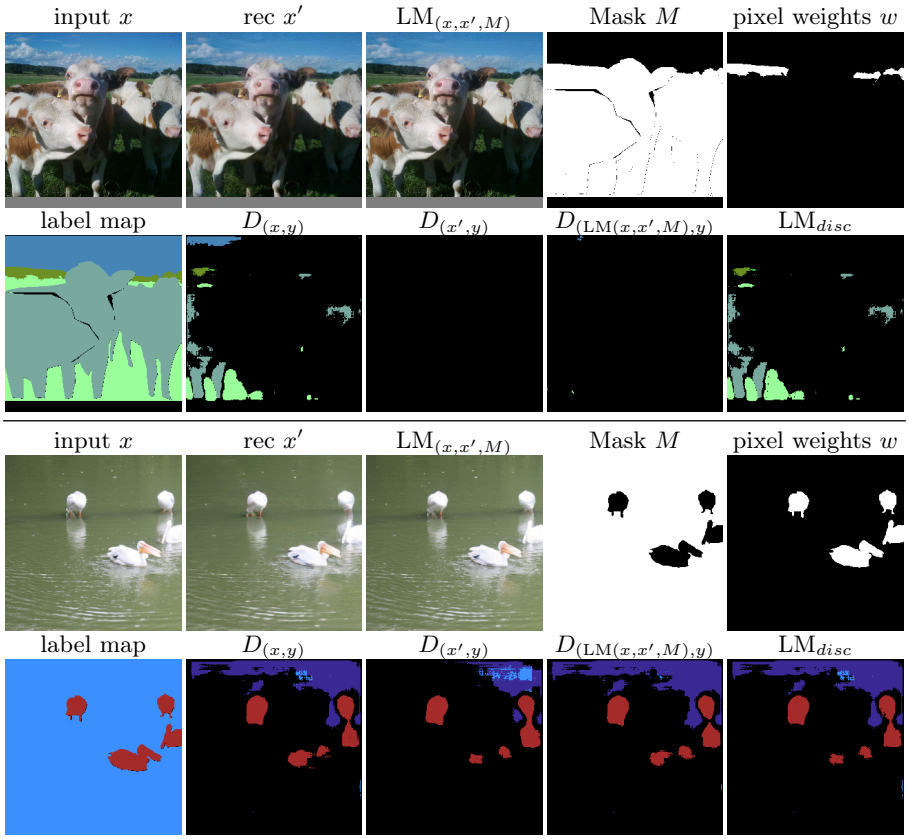

Fig. 18: Visualizing the discriminator loss components at step 1M (top row) and at an early training stage (160k, bottom row) on the Coco2017 dataset. See Fig. 17 for a detailed description.

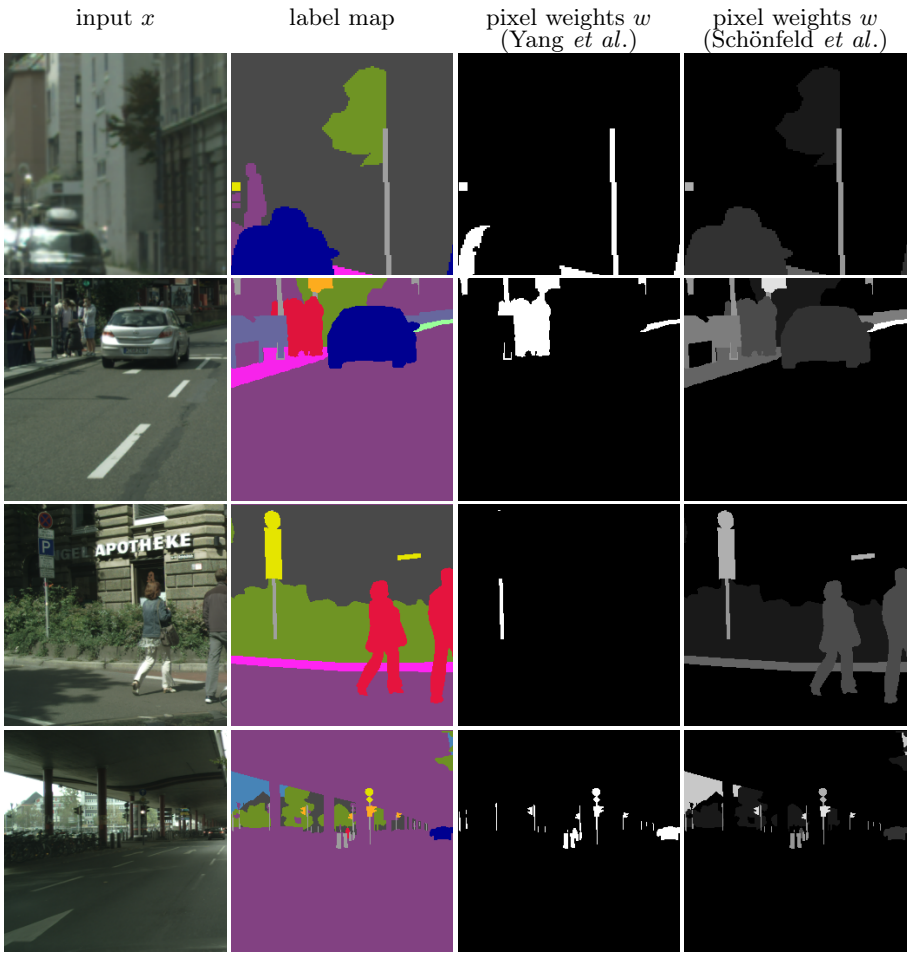

Fig. 19: Comparing pixel weighting schemes based on instance size (third column, Yang *et al.*, 2019) and class imbalance (fourth column, Schönfeld *et al.*, 2021). We map the pixel weights w to a pre-defined color map for better visualization; the brighter the color, the larger the weight. Note that in Schönfeld *et al.* (2021) identical class segments share the same pixel weights (same color).

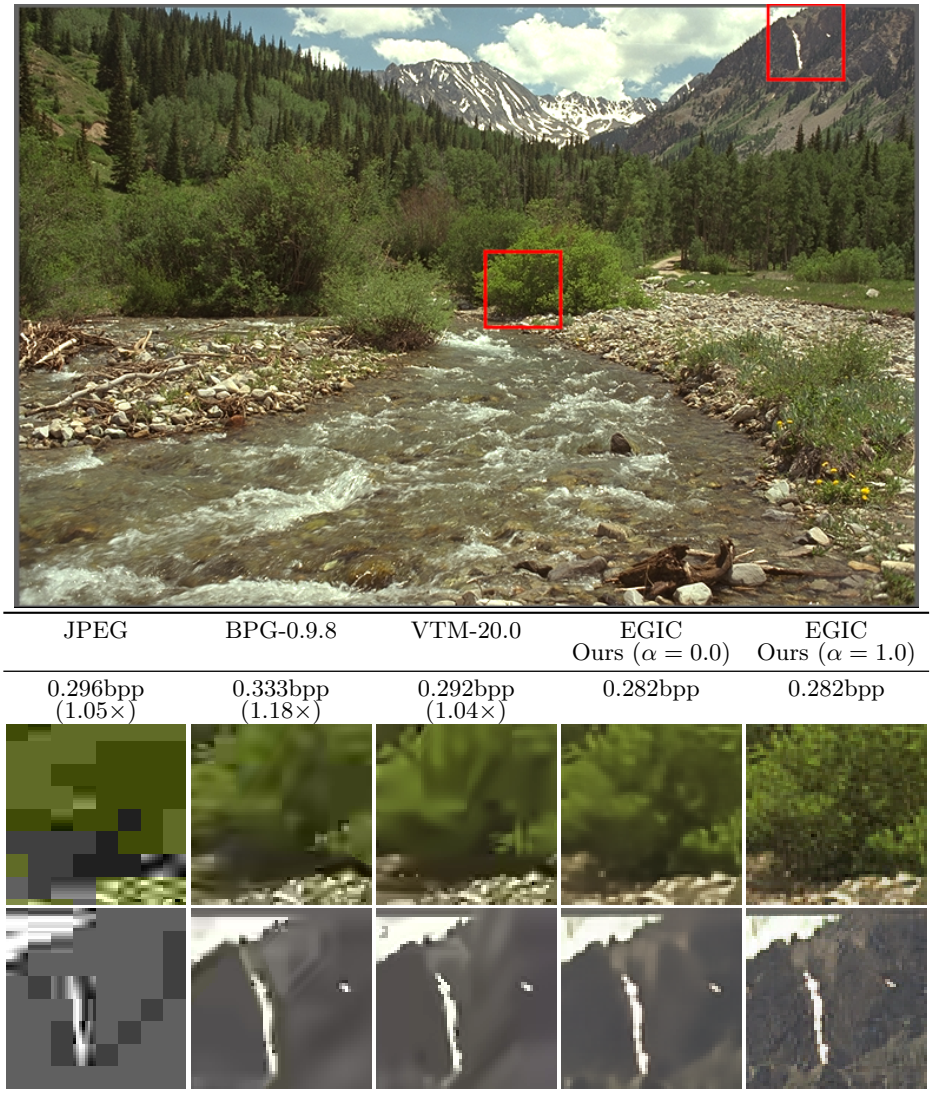

Fig. 20: Visual comparison of EGIC ($\alpha = \{0.0, 1.0\}$) with JPEG, BPG-0.9.8 and VTM-20.0 on the Kodak dataset (kodim13).

Fig. 21: Visual comparison of EGIC ($\alpha = \{0.0, 1.0\}$) with JPEG, BPG-0.9.8 and VTM-20.0 on the Kodak dataset (kodim22).

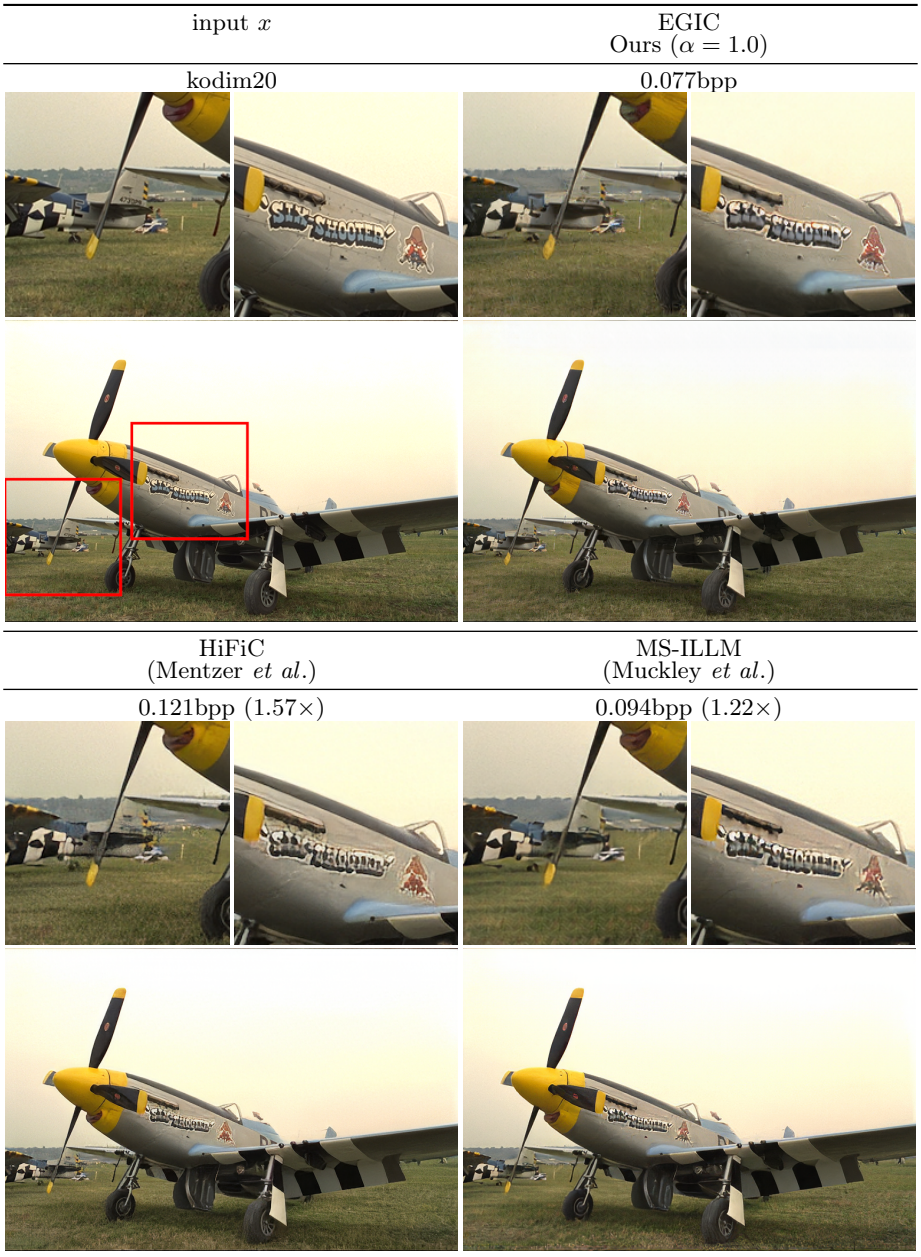

Fig. 22: Visual comparison of EGIC ($\alpha = 1.0$) with HiFiC and MS-ILLM on the Kodak dataset (kodim20). Note that our method better preserves textual information and texture (grass), despite using less bpp.

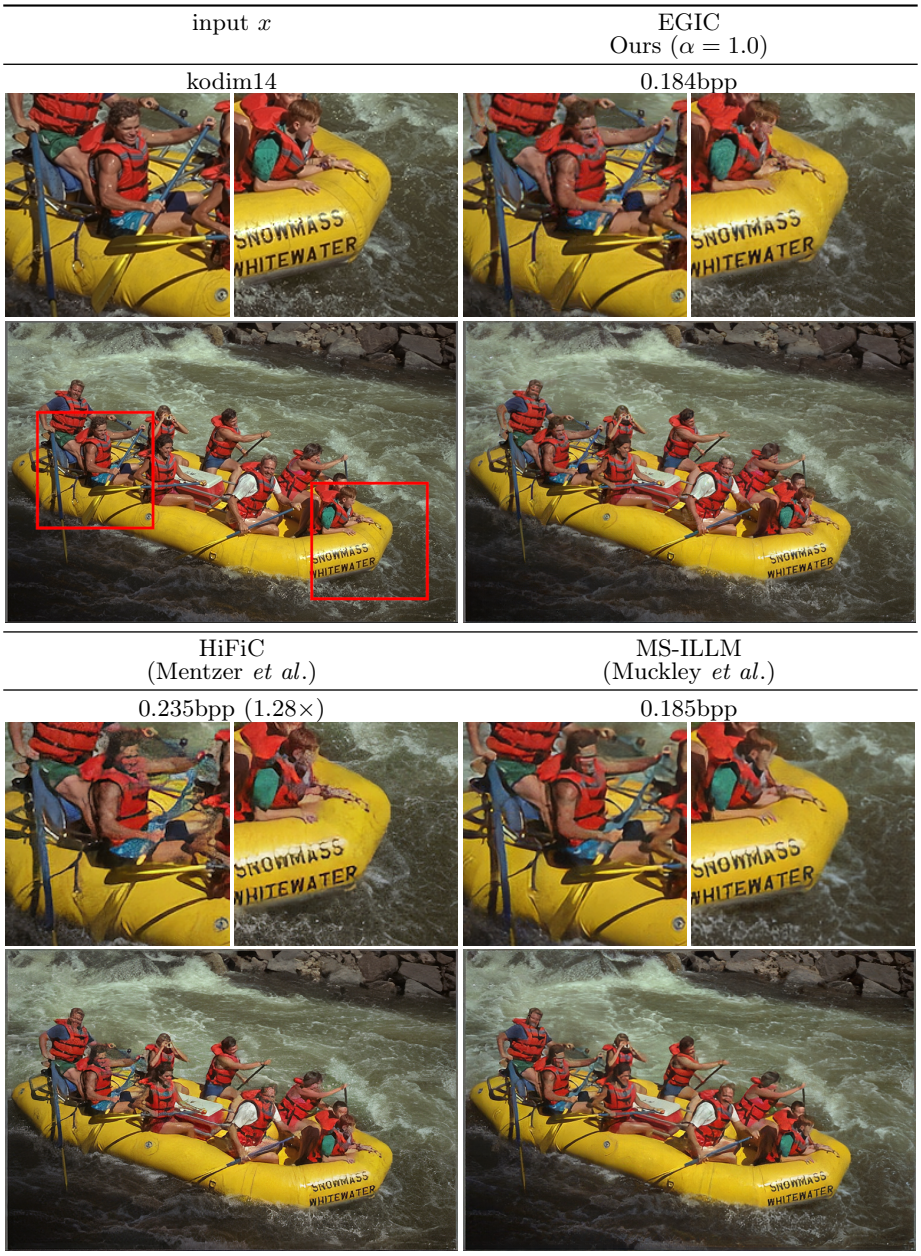

Fig. 23: Visual comparison of EGIC ($\alpha = 1.0$) with HiFiC and MS-ILLM on the Kodak dataset (kodim14). Note that our method better preserves small faces.

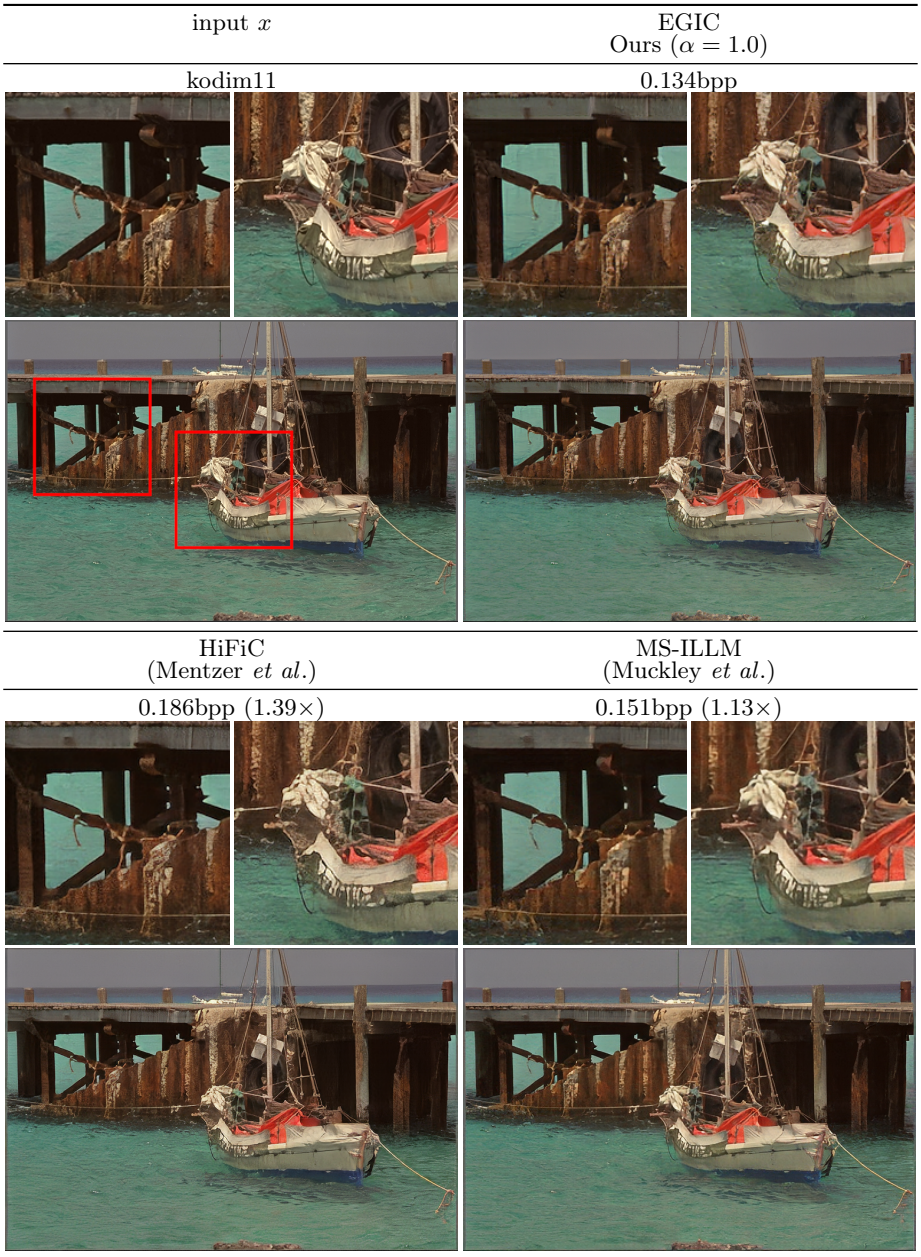

Fig. 24: Visual comparison of EGIC ($\alpha = 1.0$) with HiFiC and MS-ILLM on the Kodak dataset (kodim11). Note that our method better preserves small details (*e.g.*, the rope in the left image), despite using less bpp.

Fig. 25: Visual comparison of EGIC ($\alpha = 1.0$) with HiFiC and MS-ILLM on the Kodak dataset (kodim21). Note that our method better preserves small details (*e.g.*, the people in the left image).

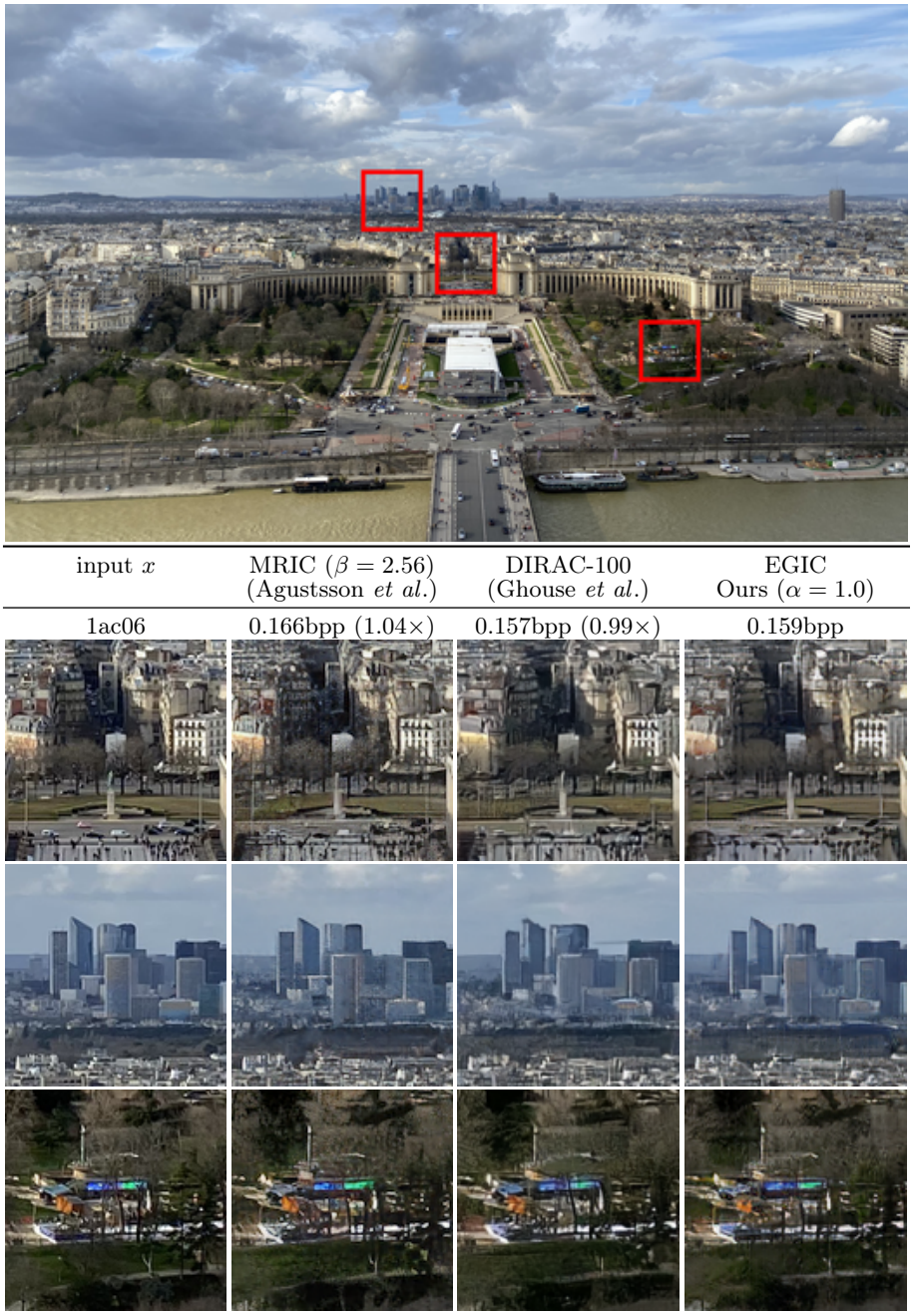

Fig. 26: Visual comparison of EGIC ($\alpha = 1.0$) with MRIC and DIRAC-100 on the CLIC dataset (1ac06). Note that EGIC has less artifacts (compared to MRIC) and better retrains color (compared to DIRAC).

Fig. 27: Visual comparison of EGIC ($\alpha = 1.0$) with MRIC and DIRAC-100 on the CLIC dataset (46c18). Note that we use less bpp.

input x	PO-ELIC (He *et al.*)	EGIC Ours ($\alpha = 1.0$)
732bf	0.068bpp (0.95×)	0.0716bpp

Fig. 28: Visual comparison of EGIC ($\alpha = 1.0$) with PO-ELIC, the winning solution of the CLIC 2022 competition, using our lowest bit-rate setting.

Fig. 29: Visual comparison of EGIC ($\alpha = 1.0$) with HFD/DDPM on the Kodak dataset (kodim24). Note that the quality of HFD/DDPM (250 steps) largely depends on the base reconstruction HFD/DDPM (ELIC).

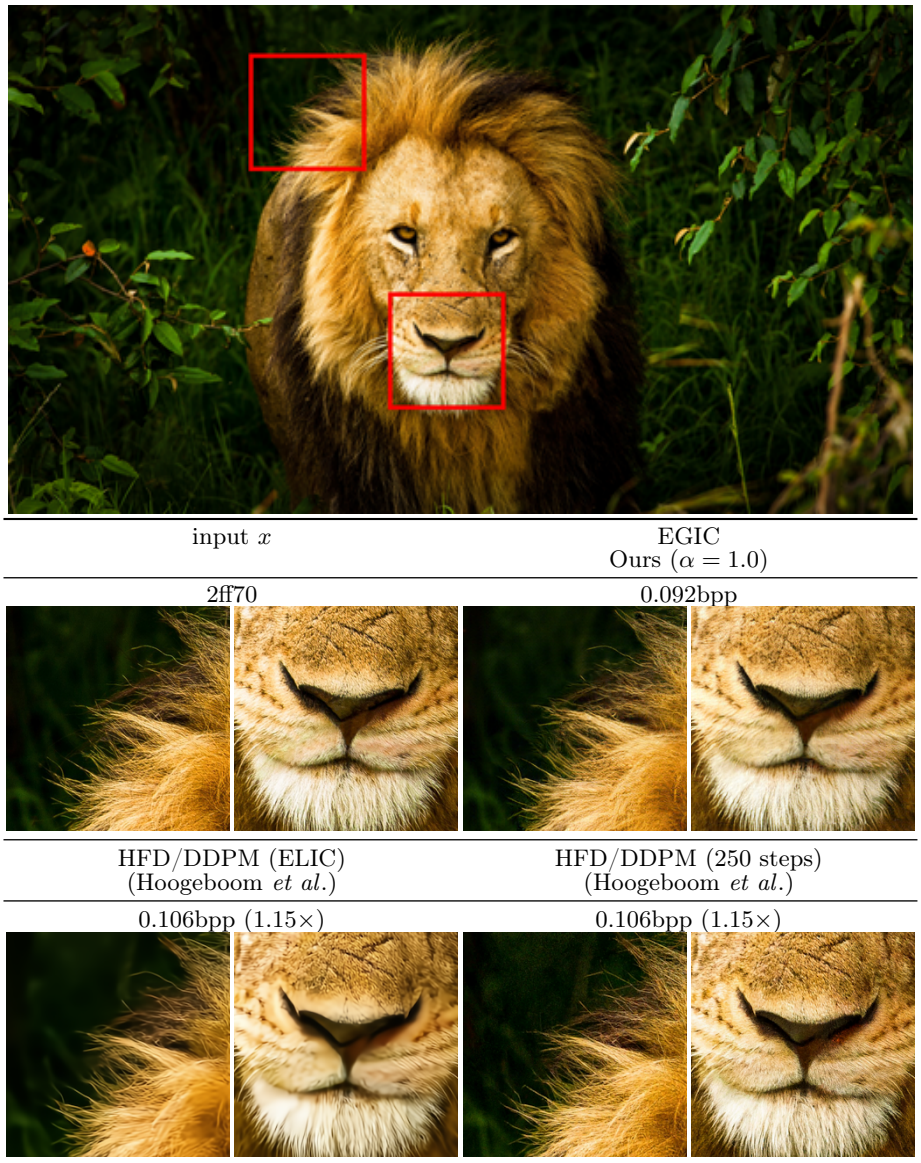

Fig. 30: Visual comparison of EGIC ($\alpha = 1.0$) with HFD/DDPM on the CLIC 2022 dataset (2ff70). We leave the assessment to the reader.