8 Supplementary Material for "Learning a Code-Space Predictor by Exploiting Intra-Image-Dependencies"

8.1 Differentiable Coding

8.1.1 Quantization

Before encoding symbols of the latent representation z, they are quantised into a discrete code z_q. We empoy a linear quantisation in the encoder:

$$z_q = \text{round}(z) \tag{14}$$

The motivation for this choice is that both the encoder and the decoder are highly nonlinear and can incorporate any transformations that increase the efficiency fo quantisation. The rounding operation's gradient however vanishes almost everywhere, which makes gradient-based optimization impossible. This can be dealt with by setting $\bar{z}_q = z + \eta$ with $\eta \sim \mathcal{U}(-0.5, 0.5)$ during training, as rounding noise can be well approximated by uniform noise.

8.1.2 Entropy Estimation

The minimization of the coding rate with respect to the parameters of the encoder requires a differentiable rate measurement with respect to the symbols z_q over which the rate is measured. Given an estimate of the code symbol probabilities $P_{\hat{z}_q}(z_q)$, the arithmetic coder can approach the cross entropy

$$H_{z_q,\hat{z}_q} = \sum_{z_q} P_{z_q}(z_q|x) \log_2(P_{\hat{z}_q}(z_q; \psi_{\hat{z}_q})) \tag{15}$$

between the estimate $P_{\hat{z}_q}$, parametrised by $\psi_{\hat{z}_q}$ and the actual probabilities $P_{z_q}(z_q|x)$ given the image x. In eq. 15, the probability of discrete symbols is required, which does not suit the continuous approximation of the quantisation as described above. Replacing the actual cross entropy with the empirical one yields

$$\hat{H}_{z_q,\hat{z}} = \frac{1}{|Z_q|} \sum_{z_q \in Z_q} \log_2(P_{\hat{z}_q}(z_q; \psi_z)) \tag{16}$$

so that only the derivative $\frac{\partial P_{\hat{z}_q}}{\partial z_q}(z_q)$ needs to exist. The probability of a symbol z_q can be computed as

$$P_{\hat{z}_q}(z_q; \psi_z) = \int_{z_q - \frac{1}{2}}^{z_q + \frac{1}{2}} p_{\hat{z}}(t; \psi_z) dt = \mathcal{U}\left(z_q - \frac{1}{2}, z_q + \frac{1}{2}\right) * p_{\hat{z}}(z; \psi_z) \tag{17}$$

By the law of addition of two independent random variables, we have

$$\bar{z}_q = z + \eta \sim p_{\hat{z}} * \mathcal{U}(-0.5, 0.5) = p_{\bar{z}_q} \tag{18}$$

which leads to $P_{\hat{z}_q}(z_q; \psi_z) = p_{\bar{z}_q}(\bar{z}_q; \psi_z)$. Note that while the CDF and the PDF correspond to different distributions, they share the same parameters. At training time, $P_{\hat{z}_q}$ in eq. 16 is replaced by $p_{\bar{z}_q}$, which can be optimized using gradient descent and suits the rounding error approximation. During testing, the estimates for the symbols of z_q are given by eq. 17.

By replacing eq. 15 with the expected cross entropy, eq. 2 can be reformulated as a probabilistic loss in terms of the Rate R:

$$\mathcal{L}_{Prob} = \frac{1}{NC}\sum_{i}^{N}\sum_{c}^{C}\log_2(p_{\tilde{z}_q}(z_{c,i}|\psi_{z,c})) + \lambda\mathbb{E}_x[D(x,\hat{x})]$$
$$= \mathbb{E}_x[\log_2(p_{\tilde{z}_q}(f_e(x;\theta_e)|\psi_{z,c}))] + \lambda\mathbb{E}_x[D(x,f_d(f_e(x;\theta_e);\theta_d))] \quad (19)$$

This objective is differentiable w.r.t. the parameters θ_e and θ_d as well as ψ_z if a suitable distribution is chosen and can hence be optimized by gradient descent. [5] uses a piecewise linear function to model this distribution while [21] employs a mixture of Gaussians. Our approach uses a mixture of Gaussians conditioned on the values of neighbouring pixels as described in Section 3.1.

8.2 Experiments

8.2.1 Network Architecture and Training

The two variants of the network architecture are shown in Table 1. To make our work easier to compare to other works, we chose ImageNet [10] as source of training images. Specifically, we selected those images where height and width exceed 1000px. At training time, we resized these images with a random factor uniformly chosen between 30% and 70%. Furthermore, we applied horizontal flips as data augmentation method. After this preprocessing, we extracted patches of size 128×128px at random positions. We trained all networks on mini-batches of 12 patches using the Adam-variant [12] of stochastic gradient descent. The initial learning rate was set to 10^{-4} and was reduced after 330.000 and 350.000 iterations by factor 0.2. We trained for a total of 400.000 iterations. All experiments were carried out using TensorFlow [1].

8.2.2 Conventional Codecs

We evaluate BPG with maximum compression, apply it to the YCbCr color space and do not use any chroma subsampling (options `-m 9 -c ycbcr -f 444`), which we found to be the best performing configuration in terms of MS-SSIM. For JPEG2000 we use the Kakadu [20] implementation as it outperforms OpenJPEG when using the MS-SSIM norm on the datasets used.

8.2.3 Integration

Modern conventional codecs combine a multitude of coding tools with different performance characteristics into one codec. The goal is to create a codec with higher coding gain that can choose from those coding tools whichever suits the data to be encoded best. This often involves making binary decisions and signalling those to the decoder. If one wanted to learn such binary decisions in a deep learning framework, one would have to resort to reinforcement learning. As our goal is to demonstrate that integrating to different approaches to context modelling yields even higher coding gain. For that matter, we chose a simple integration approach where the hierarchical code serves as additional features for the spatial predictor. Fig. 6 depicts a schematic overview of the encoding process of the spatial predictor reduced to a single spatial dimension for simpler representation. In the same fashion, Fig. 7 illustrates the hierarchical approach proposed in [6]. We combine both approaches by providing

Table 1: The network structure used for experiments reported in Section 4.2.

Operation	**Kernel**	**Sampling**	**Filters** (<0.9bpp)	**Filters** (>0.9bpp)
		Encoder		
Convolution	5x5	2x2 ↓	64	96
SGDN	1x1		64	96
Convolution	5x5	2x2 ↓	128	192
SGDN	1x1		128	192
Convolution	5x5	2x2 ↓	256	288
SGDN	1x1		256	288
Convolution	5x5	2x2 ↓	N	N
SGDN	1x1		N	N
		Spatial Predictor		
Convolution	2x2		256	256
ReLU				
Convolution	2x2		256	256
ReLU				
Convolution	4		2kN	2kN
Identity/Exponential				
		Decoder		
Transp. Convolution	5x5	2x2 ↑	256	288
SIGDN	1x1		256	288
Transp. Convolution	5x5	2x2 ↑	128	192
SIGDN	1x1		128	192
Transp. Convolution	5x5	2x2 ↑	64	96
SIGDN	1x1		64	96
Transp. Convolution	5x5	2x2 ↑	3	3

the spatial predictor with the output of the hierarchical decoder. The receptive field of the hierarchical component is much larger than that of the spatial predictor as subsampling is applied in its encoder and upsampling in its decoder. Hence, the hierarchical coder can serve the spatial predictor information about where and from which direction to predict. This is shown in Fig. 8.

8.2.4 Further Visual Results

Figures 9 and 10 show further visual results with heatmaps of the coding cost when applying the integrated predictor and for an independent model. For the integrated predictor, the average rate lies at about 0.6 bits/pixel. From the heatmaps it's clear that while regions with complex content remain difficult to encode, the coding cost is significantly lowered for regions which are easily predictable such as sky, skin or surfaces with little or only slowly varying texture.

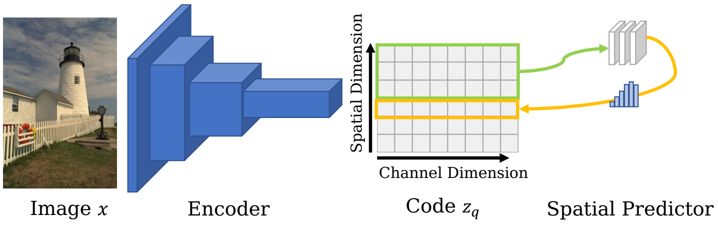

Figure 6: Schematic representation of the encoding process using our spatial predictor. The prediction is carried out only in the spatial dimension. All channels at one spatial position is prediction in parallel.

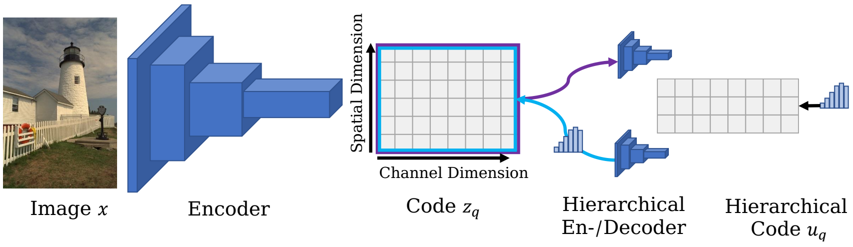

Figure 7: The encoding process in the hierarchical codec as proposed by Balle et al. [6]. There are two codes, z_q and u_q where the spatial resolution of u_q is $\left[\frac{1}{4}, \frac{1}{4}\right]$ of the resolution of z_q. There are two encoders and two decoders necessary to realise the full codec. The distribution that is used to encode u_q with the arithmetic coder is a fixed distribution.

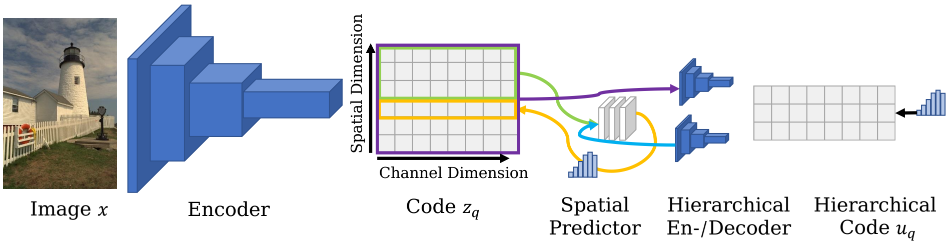

Figure 8: Our simple integration of the spatial predictor and the hierarchical codec. The hierarchical coding module is not used to estimate probability distributions of the code symbols in z_q directly but serves as additional feature input to the spatial predictor. It can hence provide higher level information to the spatial predictor as to which parts of the image are suitable for prediction and which prediction directions are the most efficient. All three parts (the original transformation, the spatial predictor and the hierarchical codec) can be trained jointly in an end-to-end fashion.

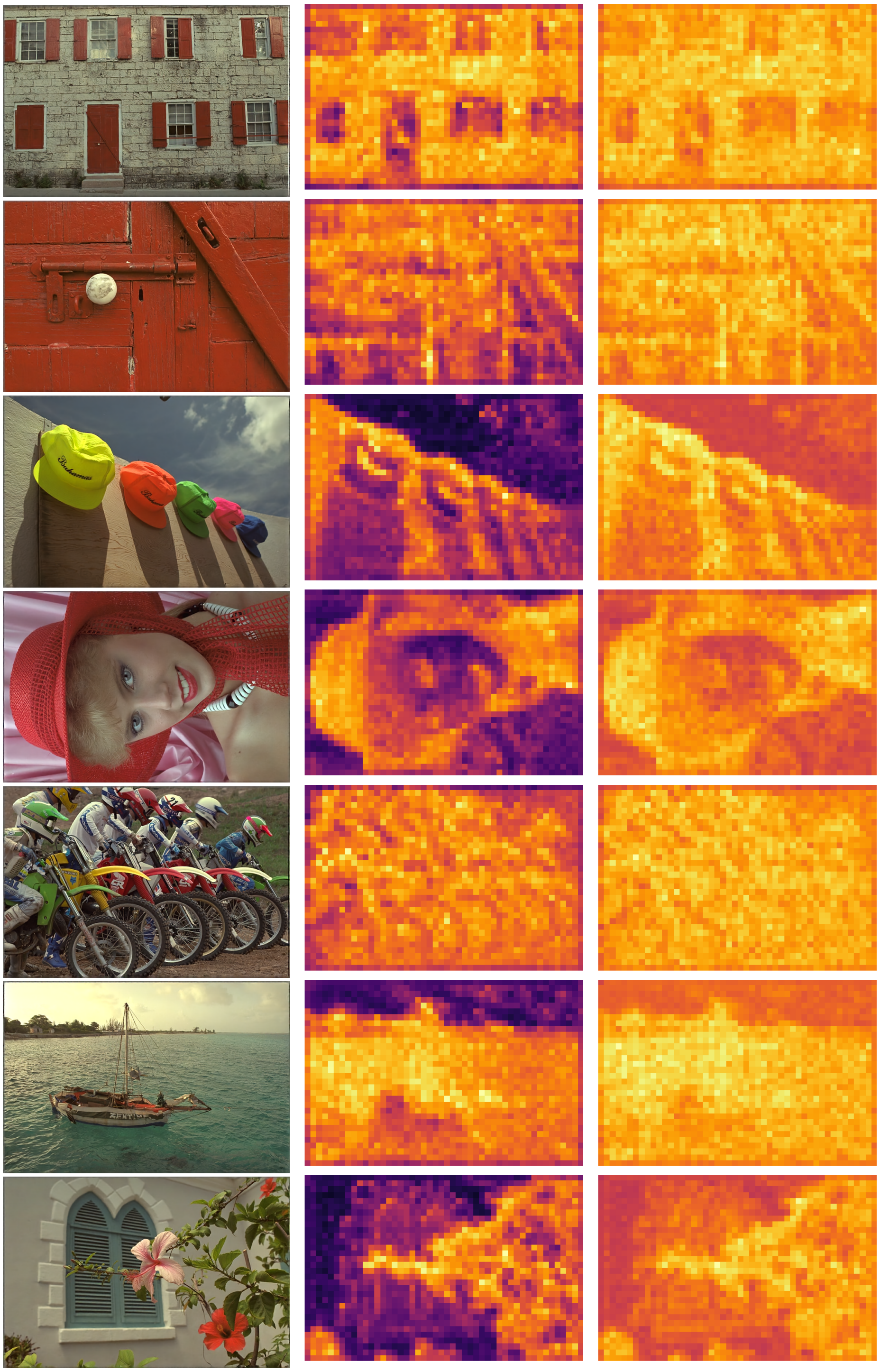

Figure 9: Reconstruction (right), spatial prediction based (center) and independent (left) coding cost spatial distribution for images 1 to 7 of the Kodak dataset.

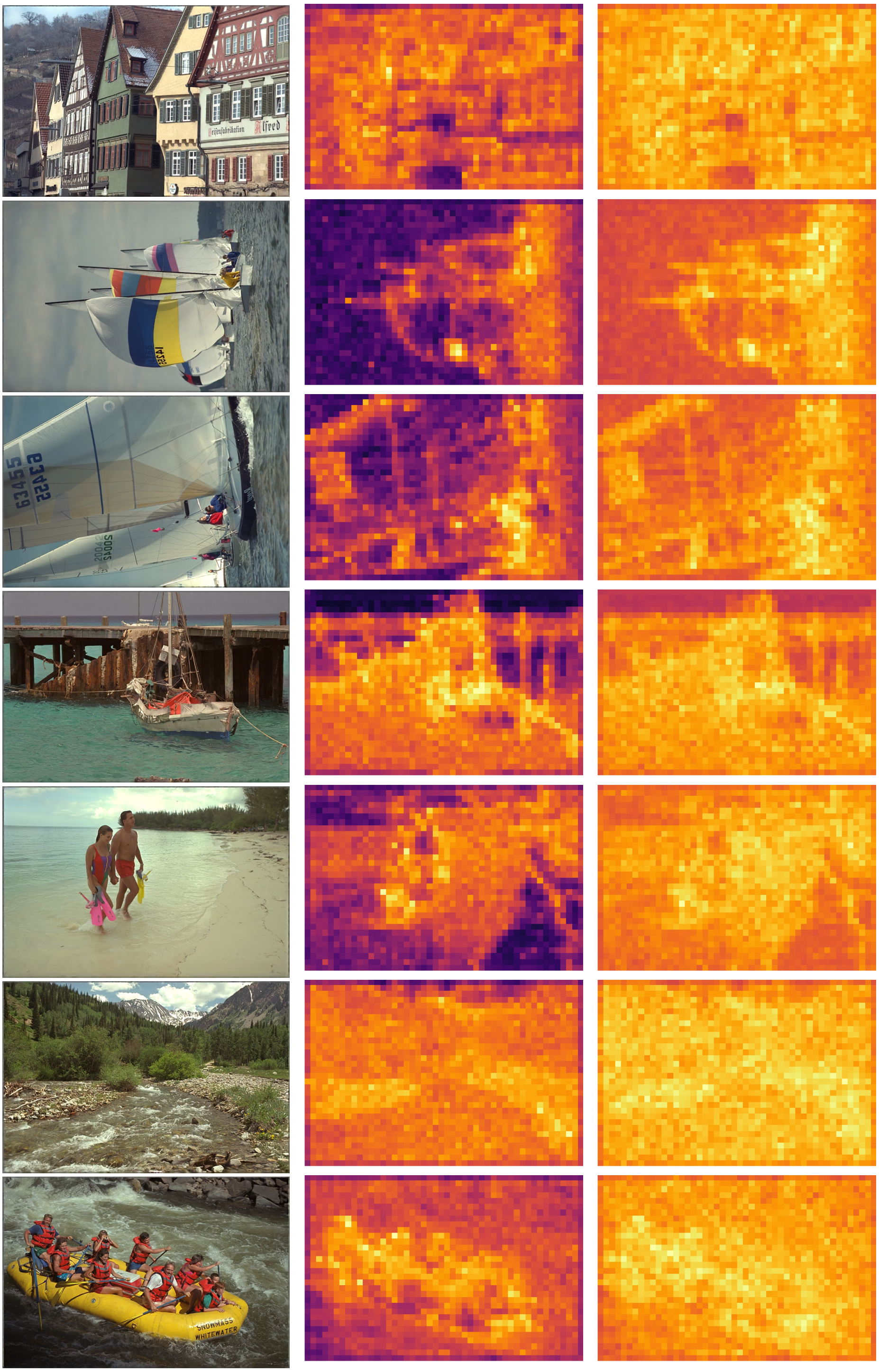

Figure 10: Reconstruction (right), spatial prediction based (center) and independent (left) coding cost spatial distribution for images 8 to 14 of the Kodak dataset.