Learning a Code-Space Predictor by Exploiting Intra-Image-Dependencies

Jan P. Klopp
klopp@media.ee.ntu.edu.tw
Yu-Chiang Frank Wang
http://vllab.ee.ntu.edu.tw
Shao-Yi Chien
http://media.ee.ntu.edu.tw
Liang-Gee Chen
http://video.ee.ntu.edu.tw

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan

Abstract

We introduce a neural network based image coding model that utilizes a code-space predictor to reduce code length by modelling dependencies within the code. Inspired by the prediction mechanism of inpainting, we learn a spatial predictor in the code space to efficiently deal with spatial dependencies. It is jointly trained with the codec to estimate a probability distribution from adjacent code symbols. The resulting code stores information related to the image and the prediction of neighbours. To improve its optimization, we adapt the Generalized Divisive Normalization into a sparse variant. The resulting model outperforms other prediction based methods. We show that when integrated with the recently proposed hyperprior model, our approach obtains state-of-the-art performance for CNN-based image codecs on the MS-SSIM scale.

1 Introduction

The continuous growth of internet traffic and storage needs has mostly been driven by image and video content, which has drawn significant interest in new image compression methods from the machine learning community [2, 3, 4, 5, 6, 11, 14, 15, 17, 21, 22, 23]. Their operating scheme is similar to that of traditional codecs [7, 16, 18] which transform an image x into a representation z and apply quantisation before encoding the discrete representation. These transformations are hand designed and often based on the separation of frequency spectra. On the other hand, machine learning based approaches can adopt their transformations to the distribution given by training data and thereby allow simple fitting to new compression scenarios.

Conventional codecs use prediction schemes, such as block-wise prediction in HEVC [19], to improve coding gain beyond transform coding. In such schemes, a decoded block can be used to predict an adjacent block, controlled by information that creates additional overhead. Like prediction via inpainting [3], these methods operate in the spatially large

pixel space, which makes them expensive to implement in a convolutional neural network. Hence, another way is to predict in the spatially more dense latent space z. Typically, the quantization result of z is encoded using an entropy coder, such as an arithmetic coder, that simply assumes statistical independence between code symbols. Using spatial prediction such dependencies are explicitly modelled to increase coding gain. We train a spatial prediction architecture that can be efficiently applied in the code domain. It provides the arithmetic coder with a neighbourhood-based probability estimate for the value to be encoded and is jointly optimized with the codec. This leads to an encoder being able to embed predictive information about neighbouring code symbols alongside transform-domain content of the actual image. To facilitate its optimization, a variant of Generalized Divisive Normalization transform with certain sparsity properties is proposed. Augmenting a simple feed-forward model with the spatial predictor yields competitive performance, a new state-of-the-art in learned image coding is reached once a hierarchical coding approach is integrated.

In summary, our contributions are as follows:

- A spatial code-space predictor is introduced to model dependencies between neighbouring code variables and reduce the code length, leading to demonstrated coding gain over previous methods.
- Additional improvement by integrating the spatial predictor and a hierarchical coding approach is shown.
- A sparse version of the Generalized Divisive Normalization to improve coding performance is introduced.

2 Review of Learned Image Compression

Image coding is typically implemented as a variant of transform coding [9] where the input image x undergoes an encoding transformation $f_e(x) = z$ into code z before being quantised into $z_q = q(z)$ by some quantization function $q()$. The discrete code z_q can then encoded into a bit stream by an arithmetic coder that in the limit approaches the cross entropy

$$H_{z_q,\hat{z}_q}(x) = \mathbb{E}_{z_q}\left[\log_2\left(P_{\hat{z}_q}(z_q;\psi_{\hat{z}_q})\right)|x\right] \tag{1}$$

between the empirical code distribution $P_{z_q}(z_q|x)$ of the image and its estimate $P_{\hat{z}_q}(z_q;\psi_{\hat{z}_q})$, where the parameters $\psi_{\hat{z}_q}$ are obtained over a representative dataset. To reduce the entropy and thereby the coding rate, the encoder f_e and the estimate $P_{\hat{z}_q}$ need to be fit. At the same time, a distortion $D(x,\hat{x})$ measuring the dissimilarity between the input image x and the decoded image $\hat{x} = f_d(z_q)$ returned from the decoder f_d has to be minimized. This leads to the rate-distortion objective

$$\mathcal{L}_{RD} = H_{z_q,\hat{z}_q} + \lambda D \tag{2}$$

The parameter λ determines the exact position of the rate-distortion trade-off. When the transformations $z = f_e(x;\theta_e)$ and $\hat{x} = f_d(z_q;\theta_d)$ are chosen to be parametric, the codec can be optimized towards the choice of distortion measure and the underlying distribution of the image data. This allows to obtain codecs for certain kinds of imaging data and different applications in an entirely unsupervised fashion. To optimize the codec with gradient descent methods, all parts need to have a derivative. In the current formulation, the quantization and the entropy estimation involve discrete data and are hence not continuously differentiable.

Both can be remedied by introducing uniform noise at training time as shown for example in [5]. For brevity, the details can be found in Section 8.1 of the supplementary material.

When f_e and f_d are realized by neural networks, minimizing Eq. 2 becomes formally equivalent with training a variational autoencoder (VAE; Kingma et al. [13]) as shown by Balle et al. [5] and Theis et al. [21]. This basic framework enables us to learn more advanced image coding methods in an end-to-end manner.

3 Proposed Method

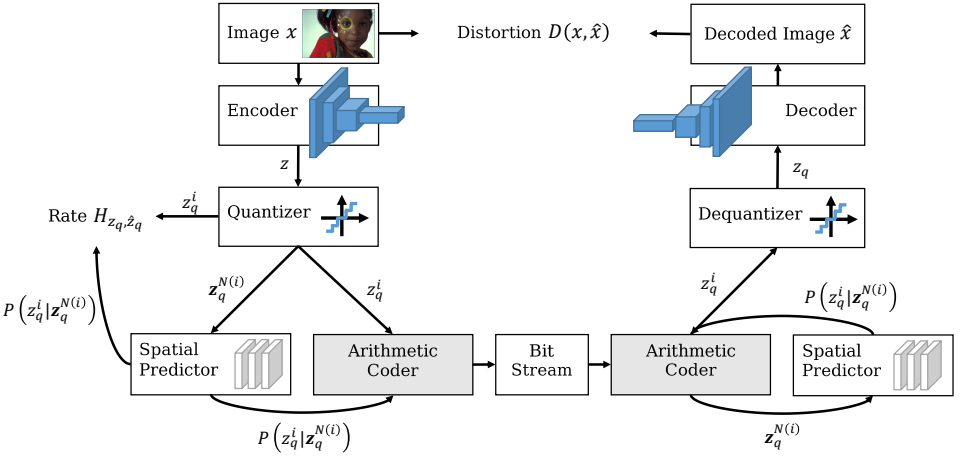

Figure 1: The encoding (left column) and the decoding (right column) process in the compressive autoencoder (CAE) augmented with the spatial code-space predictor.

3.1 Code-Space Predictor

The model of the latent code introduced in Section 2 assumes that code symbols z_q are independent of each other, i.e.

$$P_{\hat{z}_q}(\mathbf{z}_q) = \prod_i P_{\hat{z}_q}(z_q^i) \tag{3}$$

with i indicating the i-th position in the code tensor $\mathbf{z}_q \in \mathbb{Z}^{H\times W\times C}$. Balle et al. [5] have shown how to employ a special type of non-linearity to increase independence in the channel dimension of the code. Independence in the spatial domain is hard to achieve. If one patch is repeated several times, then the receptive field for the code vector corresponding to that patch would have to span all patches. This would yield a dramatic increase in complexity. Instead, we approach this problem by introducing a spatial predictor that is supposed to learn, depending on the code vector for a patch, when such neighbouring relations are likely. It thereby models dependencies between adjacent code elements by predicting the parameters ψ_p of a distribution

$$P(\mathbf{z}_q) = \prod_i P(z_q^i|\{z_q^j\}_{j\in N(i)};\psi_p) \tag{4}$$

conditioned on a neighbourhood $N(i)$ of already decoded code elements $\{z_q^j\}_{j\in N(i)}$ around the i-th position. For brevity, we denote the set of neighbourhood code elements as $\mathbf{z}_q^{N(i)} = \{z_q^j\}_{j\in N(i)}$. Fig. 1 shows the integration of the spatial predictor in the compressional autoencoder (CAE) architecture. We model the distribution in Eq. 4 as mixture of equally weighted Gaussians such that the free parameters to be predicted are the mean μ_k and the standard deviation σ_k of each component k:

$$P_{\hat{z}_q}(z_q^i|\mathbf{z}_q^{N(i)};\psi_p) = \frac{1}{K}\sum_k^K \mathcal{N}\left(\mu_k(\mathbf{z}_q^{N(i)};\psi_p), \sigma_k(\mathbf{z}_q^{N(i)};\psi_p)\right) \tag{5}$$

That is, there are $2k$ prediction functions $\mathcal{F}_p = \{\mu_k(\mathbf{z}_q^{N(i)}), \sigma_k(\mathbf{z}_q^{N(i)})\}_k$ required that. We learn these using a variational approach. Note that the predicted probabilities are conditioned on quantized, already decoded code elements which ensures that encoder and decoder both obtain identical predictions. As shown in Fig. 2, these probabilities are then provided to the arithmetic coder to ensure that the number of bits per symbol approaches the conditional cross entropy

$$H_{z_q,\hat{z}} = \sum_{z_q} P_{z_q}(z_q|x)\log_2(P_{\hat{z}_q}(z_q^i|\mathbf{z}_q^{N(i)};\psi_p)) \tag{6}$$

Using the techniques described in Section 8.1 of the supplementary material, this is replaced by a differentiable expression. The total objective to be minimized with respect to θ_e, θ_d, and ψ_p is

$$\mathcal{L}_{\text{Predictor}}(\theta_e,\theta_d,\psi_p) = \mathbb{E}_{z_q}\left[\log_2(P_{\hat{z}_q}(z_q^i|\mathbf{z}_q^{N(i)};\psi_p))\right] + \lambda\mathbb{E}_x\left[D(x,\hat{x})\right] \tag{7}$$

The objective is fully differentiable with respect to the parameters to be learned and hence gradient descent methods can be applied to minimize in an end-to-end manner.

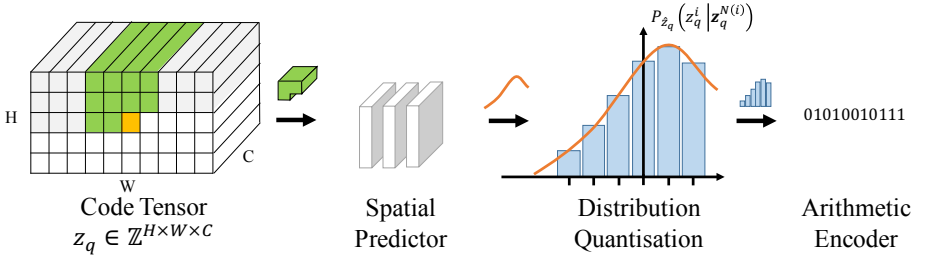

Figure 2: Process of obtaining probability distribution over a code vector (yellow) from adjacent code vectors (green). The spatial predictor takes a set of neighbouring code vectors as output and generates a probability distribution for each element in the adjacent code vector. From these distributions, probabilities for each symbol are generated and provided to the arithmetic coder.

3.2 Sparse Generalized Divisive Normalization to Ease Optimization of the Augmented CAE

Optimizing the proposed model leads to a gradient being backpropagated from the loss in Eq. 2 to z_q through three paths: via the distortion, directly through the entropy and indirectly

through the spatial predictor (see Figure 1, Quantizer in the left column), resulting in a four term total differential:

$$\frac{d\mathcal{L}_{RD}}{dz_q} = \frac{dD(x,\hat{x})}{dz_q} + \frac{\partial H_{z_q,\hat{z}_q}}{\partial z_q} + \frac{\partial H_{z_q,\hat{z}_q}}{\partial \mu_k}\frac{d\mu_k}{dz_q} + \frac{\partial H_{z_q,\hat{z}_q}}{\partial \sigma_k}\frac{d\sigma_k}{dz_q} \tag{8}$$

To ease optimization, the weight updates from each of the four terms should interfere as little with each other as possible. One way to achieve this is orthogonality by sparsity, i.e. the sets of weights that are altered by each of the gradients are mutually exclusive. At the same time, we would like to preserve the beneficial effects to transform coding of the Generalized Divisive Normalization (GDN) transform as introduced in [4] and first applied to image coding networks in [5]. The GDN aims at reducing redundancies between channels and hence applies a different normalisation to each channel at the same spatial position. The original formulation for activations $x_{i,c}$ with i being the spatial and c being the channel index is given by

$$\mathrm{GDN}(x_{i,c}) = \frac{x_{i,c}}{(\beta_c + \sum_e^C w_{c,e}|x_{i,e}|^{\alpha_{c,e}})\varepsilon_e} \tag{9}$$

Its derivative w.r.t $x_{i,d}$ is given by

$$\frac{\partial \mathrm{GDN}(x_{i,c})}{\partial x_{i,d}} = \frac{\delta_{c,d}}{(\beta_c + \sum_e^C w_{c,e}|x_{i,e}|^{\alpha_{c,e}})\varepsilon_c} - \frac{\alpha_{i,d} w_{c,d}\varepsilon_c x_{i,c}|x_{i,d}|^{\alpha_{i,d}-1}\mathrm{sign}(x_{i,d})}{(\beta_c + \sum_e^C w_{c,e}|x_{i,e}|^{\alpha_{c,e}})\varepsilon_c+1} \tag{10}$$

By inserting a ReLU non-linearity in the denominator, we can achieve a sparser gradient. We further simplified the function by setting $\alpha_{i,j} = 2$ and $\varepsilon_i = 0.5$. In addition, β_c is replaced by $0.1 + \mathrm{ReLU}(b_c)$, which, if b_c is initialized to 0.9 leads to a stable convergence behaviour. The resulting sparse GDN is shown in eq. 11.

$$\mathrm{SGDN}(x_{i,c}) = \frac{x_{i,c}}{\sqrt{0.1 + \mathrm{ReLU}(b_c) + \mathrm{ReLU}(\sum_e^C w_{c,e}x_{i,e}^2)}} \tag{11}$$

Its derivative w.r.t $x_{i,d}$ is given by

$$\frac{\partial \mathrm{SGDN}(x_{i,c})}{\partial x_{i,d}} = \frac{\delta_{c,d}}{\sqrt{0.1 + \mathrm{ReLU}(b_c) + \mathrm{ReLU}(\sum_e^C w_{c,e}x_{i,e}^2)}} - \frac{w_{c,d}x_{i,d}x_{i,c}\mathbf{1}_{\sum_e^C w_{c,e}x_{i,e}^2>0}}{(0.1 + \mathrm{ReLU}(b_c) + \mathrm{ReLU}(\sum_e^C w_{c,e}x_{i,e}^2))^{\frac{3}{2}}} \tag{12}$$

Note that the first terms in both eq. 10 and eq. 12 differ from 0 only if the gradient w.r.t to the same channel c is taken, i.e. $d = c$. The second term in eq. 12, however, differs from 0 if the scalar product $\sum_e^C w_{c,e}x_{i,e}^2$ exceeds 0, the chance which, for w initialized with a distribution symmetric around 0, is 0.5, assuming that $x_{i,e}$ is independent and identically distributed over index e. Comparing this to eq. 10, the insertion of the ReLU operation induces sparsity into the GDN, which empirically leads to a higher coding gain. The sparse inverse GDN (SIGDN) is defined analogously to the inverse GDN in TensorFlow:

$$\mathrm{SIGDN}(x_{i,c}) = x_{i,c}\sqrt{0.1 + \mathrm{ReLU}(b_c) + \mathrm{ReLU}\left(\sum_e^C w_{c,e}x_{i,e}^2\right)} \tag{13}$$

4 Experimental Results

4.1 Setup

The encoder f_e and the decoder f_d are both realised as convolutional neural networks. We use four convolutional layers with SGDN/SIGDN as non-linearities for encoder and decoder. The decoder uses transposed convolutions to allow for efficient upsampling. The spatial predictor has three layers. The first two layers have 2×2 filters, the last one has a 3×3 filter, of which the bottom row and the two rightmost elements of the middle row are masked, so that no code that has not been decoded can be accessed. The two preceding layers are padded accordingly for the same reason. For the kN mean estimates of the Gaussian mixture in Eq. 5, we use the outputs of the last layer directly; for the standard deviation estimates, we use the exponential activation function to constrain them in the positive domain and avoid a collapse where the probability is only assigned to a very small fraction of the code symbols which could cause a very high entropy. In all experiments, we set $k = 4$, as we found that higher values did not result in a better coding gain. To account for the fact that higher bit rates require more filters, we chose two model configurations: a more efficient one for rate points below 0.9bpp and a more complex one for those above. Network structure and training methodology are detailed in Section 8.2.1 of the supplementary material. In order to obtain different rate-distortion trade-offs, most approaches resort to varying the parameter λ in eq. 2. For the spatial predictor, we found that varying the number of channels in the code tensor gave better performance. The reason may lie in the fact that the spatial predictor requires the code to store both, information about the reconstruction of the image as well as for the prediction of neighbouring code vectors. Hence, less channels with higher entropy may be more suitable.

4.2 Comparison to Existing Works

We compare our work to conventional codecs as well as CNN-based image coding solutions on the Kodak dataset, which is a widely adopted benchmark for image compression. It comprises 24 RGB images sized 768×512 pixel. MS-SSIM [24] is used as similarity metric as most other works have adopted it. It computes values in the interval $[0, 1]$ for two given images, where the maximum denotes maximal similarity. To obtain a rate-distortion curve, we adopt the procedure used in [17]: for all images, MS-SSIM measurements are interpolated at the same bpp values from the test results obtained from different models. At each point, the average over all images is taken and its value converted to logarithmic scale is reported. As before, we train different models for different trade-off points, varying the numbers of code channels within 32, 48, 64, 96, 128, 192 and 256.

For comparison with CNN-based codecs, we chose the state-of-the-art [6], the approaches by Rippel and Bourdev [17] and Mentzer et al. [15] as they implement different types of predictors to reduce the code entropy and the approach by Johnston et al. [11], which uses recurrent neural networks. We chose JPEG [16], JPEG2000 [18] and BPG [7] as conventional image compression codecs to compare with. The former two are widely adopted lossy image codecs with low computational footprint. BPG is based on the x265 library for video coding and tuned to optimize the SSIM metric. The results are shown in Fig. 4, a visual comparison with BPG and [15] is provided in Fig. 3. Our approach outperforms the approaches by Rippel and Bourdev [17] and Mentzer et al. [15] for all bit rates tested, although they a

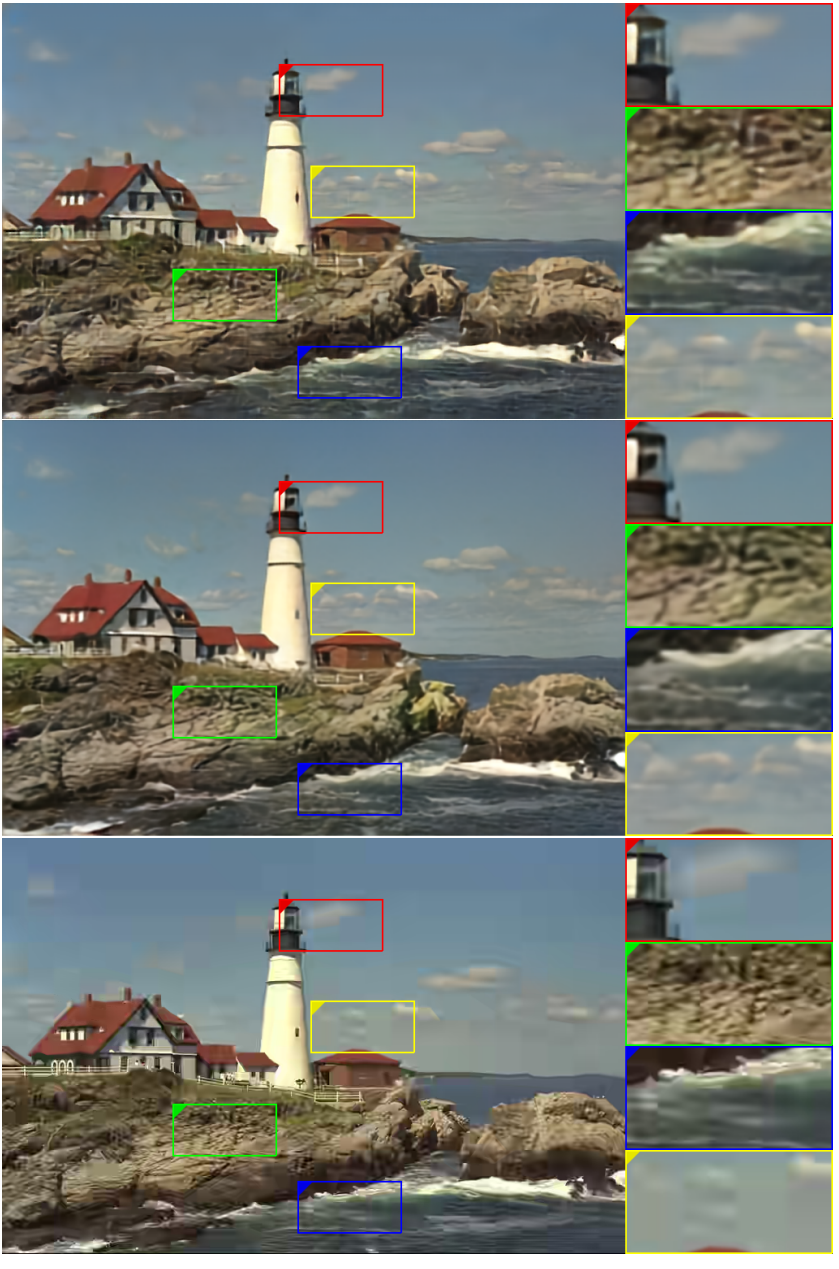

Figure 3: Visual comparison between our method (top; 0.116bpp / 0.9458 / 25.11dB), Mentzer et al. [15] (middle; 0.124bpp / MS-SSIM and PSNR n/a) and BPG (bottom; 0.113bpp / 0.9184 / 26.4dB). The first highlight (top of the lighthouse) demonstrates detail preservation of our approach compared to Mentzer et al.'s recurrent predictor. BPG has distortion and blocking artifacts (highlights two, three and four), that do not occur in both trained approaches.

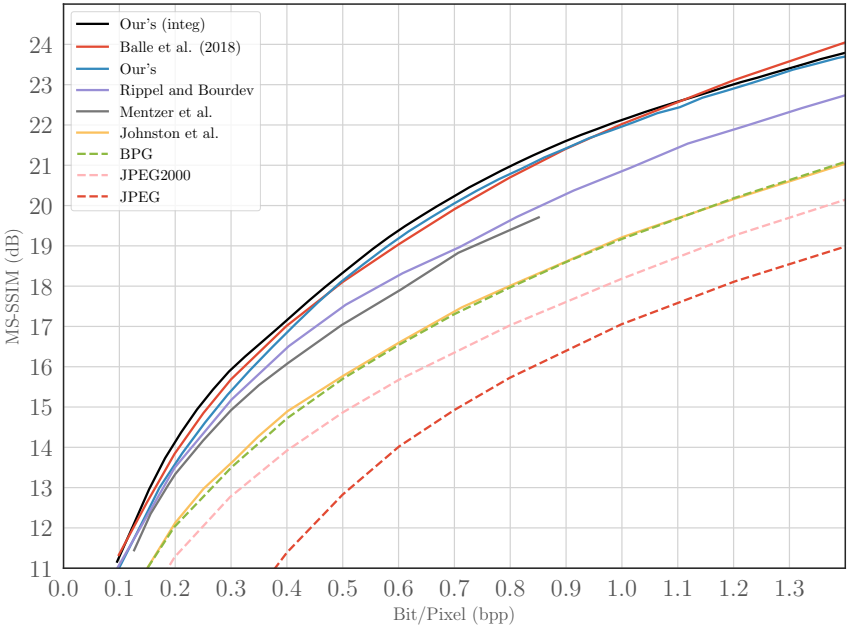

Figure 4: Comparison of coding performance on the Kodak dataset between recent empirical image coding approaches, conventional codecs and our proposed approach. MS-SSIM performance is measured in the RGB domain without chroma subsampling. BD-Rate [8] savings of our codec (with integration) against machine learned codecs are -3.2% (Balle et al.), -7.2% (Our's w/o integration), -15.9% (Rippel & Bourdev), -20.5% (Mentzer et al.) and -40.8% (Johnston et al.); against conventional codecs -41.4% (BPG), -51.4% (JPEG2000) and -65.7% (JPEG), respectively. The logarithmic MS-SSIM is computed by $-10\log_{10}(1-\text{MS-SSIM})$.

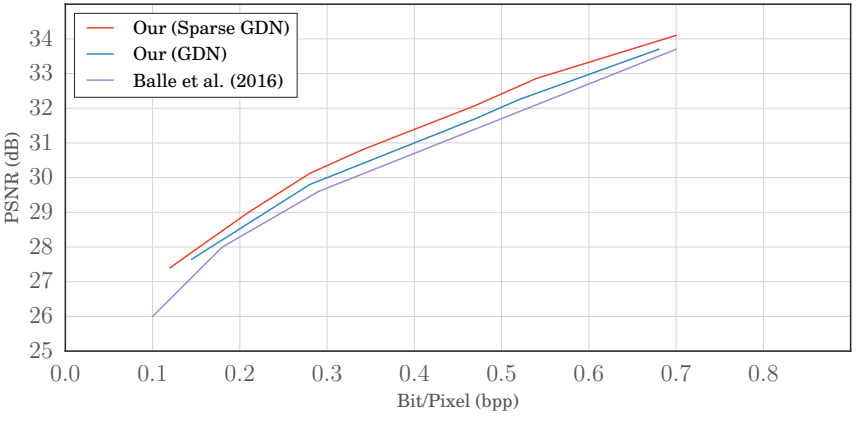

Figure 5: Comparison between [5] and our method using GDN and Sparse GDN.

employ more complex architectures. A multi scale encoder is used in the work of [17] and a much deeper 15-stage ResNet in [15]'s approach. This shows that spatial prediction is more effective than channel prediction as in [15] or bit-level prediction in [17]. Compared to the current state-of-the-art by Balle et al. [6], our approach underperforms for lower and higher bit rates, their BD-rate savings towards us are about 4%. One reason for this may be that a different network architecture is applied or that a larger, high definition custom dataset is used for training. Conventional codecs often resort to combining different coding tools to compensate for each other's weaknesses. By integrating our approach with that of Balle et al. [6] as described in Section 8.2.3 of the supplementary material, we achieve a new state of the art, thereby showing that each approach models slightly different dependencies. This state of the art achieves 3.2% higher BD-rate gain than the current best approach. We note that our reproduction of [6]'s codec does not yet give the same results as they reported, so that we assume that with all implementation details from both approaches available, an even higher coding gain is possible.

4.3 Spatial Predictor and Sparse GDN under constant Complexity

In conventional coding, complexity is often an issue. Hence we are interested in the coding gains from the spatial predictor when the complexity is kept constant. In this experiment, we show the coding gain improvement over the approach from Balle et al. [5], where code symbols are modelled as independent. We use the same network structure and only reduce the number of filters in each layer from 192 to 160 to account for the additional complexity of the spatial predictor. We vary the number of code channels for different rate-distortion trade-off points and train five different models. The results are shown in Fig. 5. Using the GDN provided by TensorFlow [1], the spatial predictor achieves a modest gain over the baseline, while the Sparse GDN yields further improvements. This shows that with approximately the same complexity in terms of floating point operations, the spatial predictor can achieve a higher coding gain.

5 Related Works

Several CNN-based image compression methods has been introduced in the literature. Baig et al. [3] have shown how to utilize inpainting to increase the coding gain, however the resulting model is complex as predictions happen in pixel space and not in the spatially sub-sampled code space. This may be the reason why their approach's overall performance is not competitive. The approaches by Rippel & Bourdev [17] and Mentzer et al. [15] combine transform coding and prediction. Rippel & Bourdev apply prediction only on the bit level and require a more complex multi-scale architecture. They further resort to an energy function to enable bit level prediction instead of training in an end-to-end manner. Mentzer et al. apply channel-wise prediction which requires them to use of a 3D-CNN and a very deep network, though [5] has shown how to reduce inter-channel dependencies with a new non-linearity. Incorporating this non-linearity allowed us to appropriately model the remaining *spatial* dependencies. The current state-of-the-art [6] uses additional latent variables to model relations between code vectors. The latent variable model acts as a scale hyperprior to the actual code, estimating the code's variance only. Though their approach is similar to ours, both approaches together achieve an even higher gain, showing that the dependencies modelled by their approach differ from those modelled in our approach. Lastly, the works

[11, 22, 23] by Johnston, Toderici et al. develop a residual coding approach, that iteratively produces layers of binary codes. Dependencies between different bit layers are modelled by a recurrent neural network, which is fundamentally different from our direct transform approach. Their approach could be augmented with a spatial predictor, though.

6 Conclusion

Our approach has shown how to successfully apply the spatial prediction concept of inpainting to the more compact code space domain. Training a spatial predictor jointly with the codec demonstrates higher coding gain than similar approaches that employ prediction. A simple integration with the hierarchical approach yields a new state of the art for CNN-based image coding, opening a path for further advances in this field.

7 Acknowledgements

We wish to thank Chih-Yao Ma for valuable discussions and feedback on a draft version of this paper and the reviewers for their comments to improve this work.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Brain. TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pages 265–284, 2016. ISBN 978-1-931971-33-1. doi: 10.1038/nn.3331. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations. *NIPS*, 2017. URL http://arxiv.org/abs/1704.00648.

[3] Mohammad Haris Baig, Vladlen Koltun, and Lorenzo Torresani. Learning to Inpaint for Image Compression. *NIPS*, 2017. URL http://arxiv.org/abs/1709.08855.

[4] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density Modeling of Images using a Generalized Normalization Transformation. *ICLR*, pages 1–14, 2016. URL http://arxiv.org/abs/1511.06281.

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end Optimized Image Compression. *ICLR*, 2017. ISSN 01973975. doi: 10.1016/S0197-3975(03)00059-6. URL http://arxiv.org/abs/1611.01704.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *International Conference On Learning Representations*, 2018. URL http://arxiv.org/abs/1802.01436.

[7] Fabrice Bellard. BPG Image Format. URL https://bellard.org/bpg/.

[8] Gisle Bjøntegaard. Calculation of average PSNR differences between RD-curves. Technical report, ITU-T SG16/Q6, Austin, Texas, USA, 2001.

[9] V.K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(September):9–21, 2001. ISSN 1053-5888. doi: 10.1109/79.952802.

[10] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206848.

[11] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks. *Computer Vision and Pattern Recognition*, 2017. URL http://arxiv.org/abs/1703.10114.

[12] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15, 2015. ISSN 09252312. doi: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

[13] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *ICLR*, pages 1–14, 2014. ISSN 1312.6114v10. doi: 10.1051/0004-6361/201527329. URL http://arxiv.org/abs/1312.6114.

[14] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning Convolutional Networks for Content-weighted Image Compression. 2017. URL http://arxiv.org/abs/1703.10553.

[15] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional Probability Models for Deep Image Compression. 2018. URL http://arxiv.org/abs/1801.04260.

[16] William B Pennebaker and Joan L Mitchell. *JPEG still image data compression standard*, volume 34. 1993. ISBN 0442012721. URL http://books.google.de/books?hl=de{&}lr={&}id=AepB{_}PZ{_}WMkC{&}oi=fnd{&}pg=PR13{&}dq=jpeg+{&}ots=UQFPcz9RmP{&}sig=yPBBc55RToO-8ORhFfTzgRj4m5s{#}v=onepage{&}q=jpeg{&}f=false.

[17] Oren Rippel and Lubomir Bourdev. Real-Time Adaptive Image Compression. *ICML*, 2017. URL http://arxiv.org/abs/1705.05823.

[18] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi. JPEG2000: The upcoming still image compression standard. *Pattern Recognition Letters*, 22(12):1337–1345, 2001. ISSN 01678655. doi: 10.1016/S0167-8655(01)00079-4.

[19] Gary J. Sullivan, Jens Rainer Ohm, Woo Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. ISSN 10518215. doi: 10.1109/TCSVT.2012.2221191.

[20] Taubman (University of New South Wales). Kakadu Software. URL http://kakadusoftware.com/.

[21] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders. *ICLR*, pages 1–19, 2017. URL http://arxiv.org/abs/1703.00395.

[22] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable Rate Image Compression with Recurrent Neural Networks. *International Conference On Learning Representations*, pages 1–9, 2015. URL http://arxiv.org/abs/1511.06085.

[23] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full Resolution Image Compression with Recurrent Neural Networks. *Computer Vision and Pattern Recognition*, 2016. ISSN 08936080. doi: 10.4135/9781412985277. URL http://arxiv.org/abs/1608.05148.

[24] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multi-scale structural similarity for image quality assessment. *IEEE Asilomar Conference on Signals, Systems and Computers*, 2:9–13, 2003. doi: 10.1109/ACSSC.2003.1292216. URL https://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1292216.