This CVPR 2020 workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Efficient Context-Aware Lossy Image Compression

Jan Xu, Alex Lytchier, Ciro Cursio, Dimitrios Kollias, Chri Besenbruch, Arsalan Zafar
Deep Render Ltd
{jan.xu}@deeprender.ai

Abstract

We present an efficient context-aware lossy image compression system to participate in the Low Rate track of the CLIC 2020 Image Compression challenge. Our method is based on an autoencoder pipeline augmented with a nested hyperprior model, a PixelCNN-based context model and an adversarial loss to remove artefacts.

1. Model Architecture

The general pipeline is inspired by [8]: our model consists of an autoencoder whose latent space is efficiently compressed by using a 2-level hyperprior, together with an autoregressive context model that acts similarly to intra-predictions in standard image codecs such as JPEG [12]. Compared to [8], we additionally use a critic model trained with a GAN objective to improve the quality of the reconstructed images.

1.1. Auto-encoder

The input image is initially processed with a neural network-based encoder, which decreases the dimensionality to force a compressed representation. The encoder network is composed of 4 convolutional layers with 3x3 kernels and 384 channels, each followed by a Padé Activation Unit (PAU) [9] (see Fig. 3). We use this activation because from preliminary experiments (Fig. 1) it was determined to produce better results over Generalised Divisive Normalisation (GDN) [1], a very popular activation in the compression literature. Additionally, the second and third layers have a stride of 2, resulting in the latent space being twice downsampled spatially. The resulting latent space has dimensions (12, H/4, W/4), where 12 is the number of channels, H is the height of the input image and W is the width.

A decoder network is trained to recover the original image from the compressed latent space. Its structure is mirrored with respect to the encoder: it contains 4 convolutional layers, where layers 2 and 3 consist of a bilinear upsampling layer followed by a convolution. The activation is again PAU.

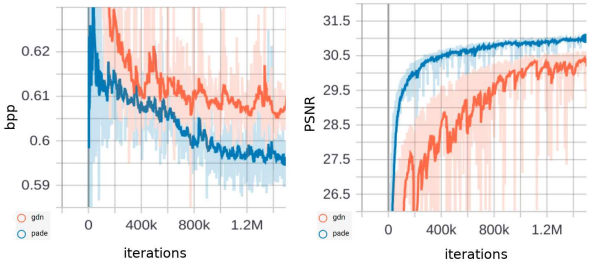

Figure 1. Validation curves for a version of our compression pipeline without context model and only 1-step hyperprior, comparing PAU (blue) vs GDN (orange) activations, ceteris paribus. The graphs show mean bpp and PSNR values over 256x256 center crops of the validation set of the Kodak dataset [6].

The latent space of this auto-encoder is further processed with two models that act in parallel: a nested hyperprior model, and an autoregressive model based on the PixelCNN++ architecture.

1.2. Hyperprior Model

The use of a hyperprior model is based on [2], where it first appears in the context of learnt image compression. Our model contains a 2-step hierarchical hyperprior, where the latent space is compressed into a hyperlatent, and the hyperlatent is compressed into a hyper-hyperlatent. Both hyperprior models have a similar structure to the main autoencoder, containing 4 convolutional layers with PAU activations and downsampling twice. All hyperprior models output parameters of a Laplacian distribution. Moreover, since the hyper-hyperlatent does not have a hyperprior, its distribution is modelled as a Laplacian with parameters learnt over the entire training dataset.

1.3. Context Model

Autoregressive models have emerged as state-of-the-art models for probability density estimation tasks [11]. Their greatest disadvantage is however the computational effort needed to run them, as they only generate a single pixel in

a forward pass. In our compression pipeline, we make use of a modified version of PixelCNN++ [10] that we named PixelCNN++Lite. Our method only considers the receptive field of the pixel being decoded in the forward pass, giving us a 10x speed up, so that the decoding time is within the maximum limits imposed by the challenge. Furthermore, compared to the original PixelCNN++ architecture we only downsample once, and use 1 layer per block as opposed to 5. Further, we substituted the Concat ELU activations with PReLU, as the original activations have a huge computational cost with only a slight performance increase over PReLU.

1.4. Quantisation Strategy

The final compressed image data consists of the latent, the hyperlatent and the hyper-hyperlatent. Further compression performance is achieved by compressing these values with an in-house range coder, and this requires them to contain only discrete values, necessitating a quantisation method to be used. We found that adding uniform noise to the latents was a good approximation of quantisation, thus we used it to "quantise" all latents during training.

1.5. Generative loss

In order to further refine the visual quality of the reconstructed image, we train a critic network and include its objective into our loss function. The critic network consists of 6 convolutional layers with 4x4 kernel sizes and leaky ReLU activations. The inputs to the critic network are the reconstructed images and the ground truth images. The loss used to train the critic network is the WGAN-GP loss [3] with lambda equal to 10. Fig. 2 shows the quality improvements obtained with the generative loss term.

1.6. Loss function

We formulate our loss function as a rate-distortion problem, where our distortion consists of two losses, the VGG perceptual loss as defined in [4] and the generator loss of WGAN-GP as defined in [3]. The rate terms consist of the cross-entropy on our latents y and z, and w. The total loss is therefore:

$$L = \lambda(D + \beta G) + R_{\hat{y}} + R_{\hat{z}} + R_{\hat{w}} \tag{1}$$

where D and G are the VGG and GAN losses respectively, balanced by the β term, while the λ term is used to control the trade-off between rate and distortion.

The rate losses for each of the latent spaces are defined below:

$$R_{\hat{y}} = -\frac{1}{H \cdot W} \sum_i log_2(p_{\hat{y}}(\hat{y}_i|\hat{z}, \theta_{hp}, \theta_{hhp}, \theta_{ctxt}, \theta_{comb})) \tag{2}$$

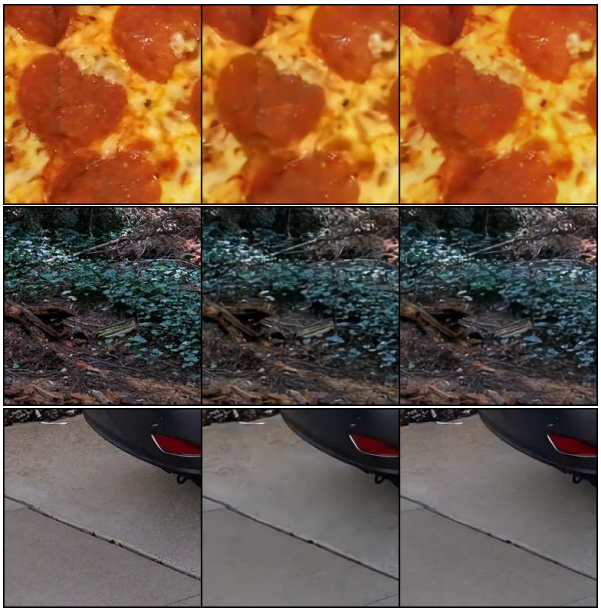

Figure 2. Reconstructed images from the CLIC validation set. Right image is the ground truth, center is with VGG loss only, left is VGG loss + GAN loss. Viewers are advised to zoom in for better clarity.

$$R_{\hat{z}} = -\frac{1}{H \cdot W} \sum_i log_2(p_{\hat{z}}(\hat{z}_i|\hat{w}, \theta_{hhp})) \tag{3}$$

$$R_{\hat{w}} = -\frac{1}{H \cdot W} \sum_i log_2(p_{\hat{w}}(\hat{w}_i|\mu_{\hat{w}_i}, \sigma_{\hat{w}_i})) \tag{4}$$

where $p_{\hat{y}}$, $p_{\hat{z}}$ and $p_{\hat{w}}$ are modelled as Laplacian distributions with mean and scale parameters, while θ_{hp}, θ_{hhp}, θ_{ctxt} and θ_{comb} represent the model parameters of the hyperprior model, the hyper-hyperprior model, the context model and the combination network (see 3), and H and W stand for height and width of the input image. In the case of $R_{\hat{w}}$ the distribution parameters $\mu_{\hat{w}}$ and $\sigma_{\hat{w}}$ are fixed parameters learnt directly during training and stored on chip during inference.

1.7. Training details

We train 8 models using the unconstrained Lagrangian cost function defined in Equation 1 with different lambda values. Each model is trained on 384x384 random crops of images from the CLIC training dataset, complemented with 10000 filtered images from the OpenImage dataset [7]. We use the Adam optimiser [5] with a learning rate of 10^{-4}. The models are trained for 150 epochs with data parallelism on 4 NVIDIA V100 GPUs, using a batch size of 4 (1 image per GPU). We apply learning rate decay, gradually reducing the learning rate to 10^{-5} over training. We do not use batch normalisation.

1.8. Conclusion

We implemented a state-of-the-art pipeline for learnt image compression, achieving a PSNR of 29.055 and MS-SSIM of 0.9502 on the validation set of the CLIC 2020 low track challenge.

References

[1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density modeling of images using a generalized normalization transformation. *CoRR*, 2015.

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Eastman Kodak. Kodak lossless true color image suite, 1993.

[7] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[8] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. *CoRR*, abs/1809.02736, 2018.

[9] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. *CoRR*, abs/1907.06732, 2019.

[10] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017.

[11] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[12] Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, Apr. 1991.

2. Appendix A

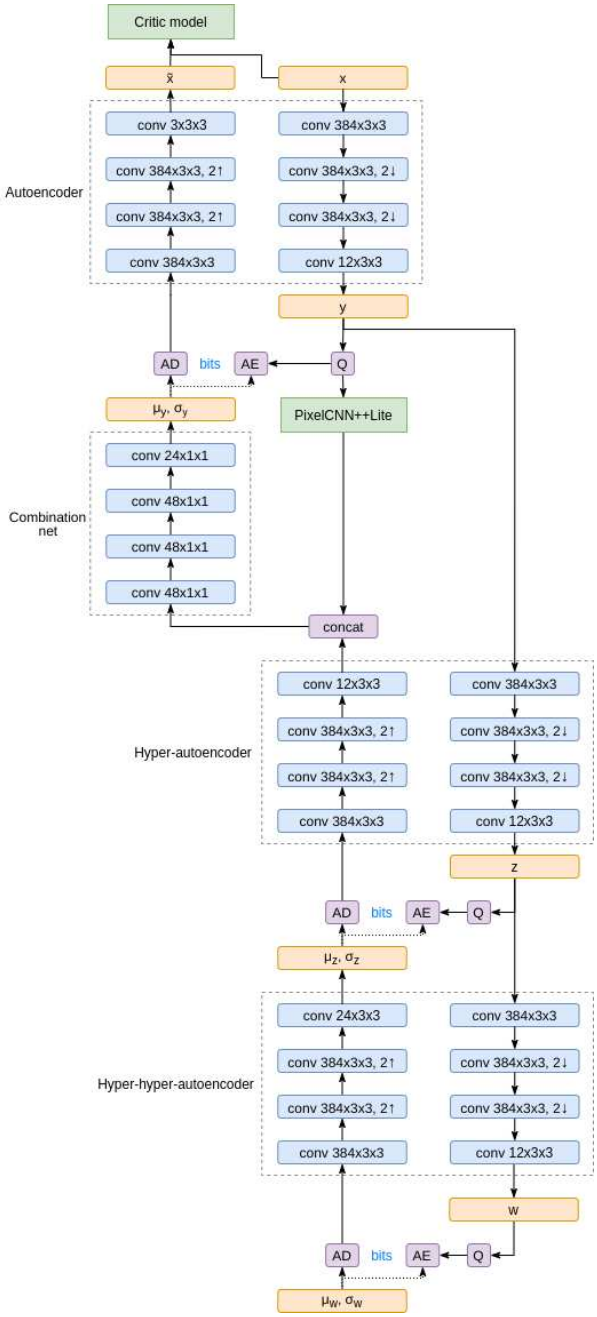

Figure 3. Architecture of compression pipeline including the two-level hyperprior and the context model. x represents the input image, y is the latent space of the main autoencoder, z is the latent space of the hyperprior and w is the latent space of the hyper-hyperprior. Q stands for quantisation, and AD and AE represent arithmetic encoder and decoder respectively. Data that is sent as bitstream has been labelled as bits. Convolutional layers are represented as (channels x kernel height x kernel width, down/upsample factor).