LOSSY IMAGE COMPRESSION WITH NORMALIZING FLOWS

Leonhard Helminger[1] **Abdelaziz Djelouah**[2] **Markus Gross**[1,2] **Christopher Schroers**[2]

[1]Department of Computer Science
ETH Zurich, Switzerland

[2]DisneyResearch|Studios
Zurich, Switzerland

ABSTRACT

Deep learning based image compression has recently witnessed exciting progress and in some cases even managed to surpass transform coding based approaches. However, state-of-the-art solutions for deep image compression typically employ autoencoders which map the input to a lower dimensional latent space and thus irreversibly discard information already before quantization. In contrast, traditional approaches in image compression employ an invertible transformation before performing the quantization step. In this work, we propose a deep image compression method that is similarly able to go from low bit-rates to near lossless quality, by leveraging normalizing flows to learn a bijective mapping from the image space to a latent representation. We demonstrate further advantages unique to our solution, such as the ability to maintain constant quality results through reencoding, even when performed multiple times. To the best of our knowledge, this is the first work leveraging normalizing flows for lossy image compression.

1 INTRODUCTION

In the recent years, several works (Toderici et al., 2016; Rippel & Bourdev, 2017; Mentzer et al., 2018; Minnen et al., 2018b) have explored the usage of deep learning for lossy image compression. These methods quickly started to challenge decades of work in transform coding and they are now at the core of a number of learning based video compression techniques such as (Lu et al., 2019; Rippel et al., 2018; Djelouah et al., 2019).

A common approach in deep lossy image compression is to map the images into a lower dimensional latent space in which a probability distribution is learned to allow for entropy coding. For this mapping, Ballé et al. (2017) proposed to leverage specialized nonlinear transformations (GDN), while Rippel & Bourdev (2017) use a pyramidal encoder to extract features of various scales. However,

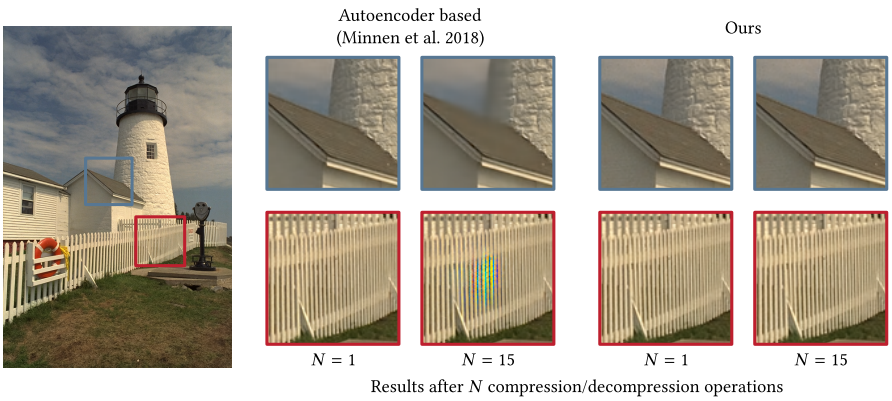


Figure 1: Using normalizing flows, image quality and bit-rate are not affected by multiple compression/decompression operations. With auto-encoder based solutions (Minnen et al., 2018a), we notice color shifts and artifacts.

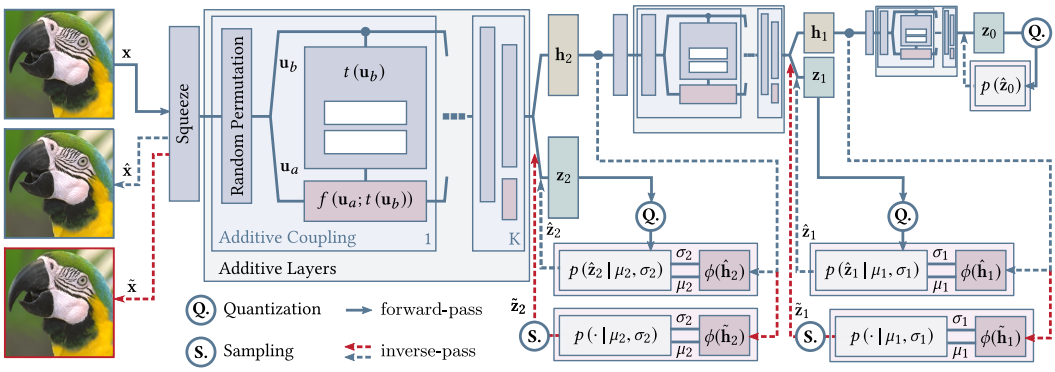


Figure 2: Overview of the flow based image compression architecture. The input image $\mathbf{x}$ is processed through a 3 level network, where each level consists of a squeeze operation (Dinh et al., 2017), followed by a series of K additive layers before a factor-out. The input image $\mathbf{x}$ is mapped to its latent representation $(\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2)$. For compression, these latents are quantized and entropy coded according to the learned distributions $p(\hat{\mathbf{z}}_2|\phi(\hat{\mathbf{h}}_2))$, $p(\hat{\mathbf{z}}_1|\phi(\hat{\mathbf{h}}_1))$ and $p(\hat{\mathbf{z}}_0)$.

the existing compression methods can typically be interpreted as variational autoencoders (VAEs), where the entropy model corresponds to the prior on the latent representation. Since autoencoders map images to a lower dimensional latent space, they impose an implicit limit on the reconstruction quality.

In this work, we propose to leverage normalizing flows as generative models in lossy image compression instead of autoencoders. By learning bijective transforms from image space to latent space, we can cover a very wide range of quality levels and effectively enable to go from low bit-rates to near lossless quality. Another interesting property of this bijective mapping is that a compressed image is always mapped to the same point. Therefore consecutive compression steps can retain the same rate-distortion performance (see Figure 1).

Summing up, the contribution of this paper is three fold: (1) To the best of our knowledge, our work is the first to explore normalizing flows for lossy image compression. (2) Our model achieves the widest range of quality levels among learned image compression methods. (3) We demonstrate additional advantages such as the capacity to maintain rate-distortion performance during multiple reencodings.

2 Lossy Image Compression with Normalizing Flows

We propose using normalizing flows for lossy image compression, and in particular we design an architecture based on additive coupling (Kingma & Dhariwal, 2018) and factor-out layers (Dinh et al., 2017). The model is illustrated in Figure 2 and it consists of three different levels bijectively mapping any input image $\mathbf{x} \in \mathcal{X}$ to its latent representation $\mathbf{z} = (\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2)$. We express lossy image compression as the quantization and the entropy coding of this latent representation.

We can obtain an estimate of the data distribution by optimizing the negative log likelihood (*nll*) for normalizing flows. Adapted to our model this equation becomes

$$\mathcal{L}_{nll}(\mathcal{X}; \theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{z}_0) + \log p_\theta(\mathbf{z}_1|\mathbf{z}_0) + \log p_\theta(\mathbf{z}_2|\mathbf{z}_1) \tag{1}$$

with all the log-determinant terms equal to 0 given that we only use additive layers. Although the distributions $p_\theta(\mathbf{z}_0)$, $p_\theta(\mathbf{z}_1|\mathbf{z}_0)$ and $p_\theta(\mathbf{z}_2|\mathbf{z}_1)$ are computed, this model is however not adapted to image compression. The reason for this is that the quantized latent values are not observed during the training as part of the dataset. This leads to our proposed image compression objective. Considering the decompressed image $\hat{\mathbf{x}}$ obtained from the rounded latents $\hat{\mathbf{z}}_l = r(\mathbf{z}_l)$

$$\hat{\mathbf{x}} = f_\theta(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2), \tag{2}$$

the objective of image compression is to both minimize the entropy (Eq.1) and the deviation from the input data point. This is expressed in the rate-distortion loss (rdl)

$$\mathcal{L}_{rdl}\left(\mathbf{x};\theta\right) = -\log p_\theta\left(\hat{\mathbf{z}}\right) + \lambda\, d\left(\mathbf{x},\hat{\mathbf{x}}\right) \tag{3}$$

with $d\left(\mathbf{x},\hat{\mathbf{x}}\right)$ a term penalizing the deviation of $\hat{\mathbf{x}}$ from the original image $\mathbf{x}$.

In order to further reduce the bit-rate, we need to exploit the models's hierarchical architecture and transmitting only a part of the latent space and sample the remaining latent values. This is illustrated in Figure 2, where on the sampling path, only latent values $\mathbf{z}_0$ are entropy coded while $\mathbf{z}_1$ and $\mathbf{z}_2$ are respectively sampled as the most likely values in the distributions $p(\mathbf{z}_1|\mathbf{z}_0)$ and $p(\mathbf{z}_2|\mathbf{z}_1)$. This new sampling path is easily included in our model through an additional reconstruction loss on the decoded sample $\tilde{\mathbf{x}} = f_\theta(\hat{\mathbf{z}}_0, \tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2)$. The final objective is then defined as follows:

$$\mathcal{L}\left(\mathbf{x};\theta\right) = -\log p_\theta(\hat{\mathbf{z}}) + \lambda\left(d(\mathbf{x},\hat{\mathbf{x}}) + d(\mathbf{x},\tilde{\mathbf{x}})\right). \tag{4}$$

Since the rounding operation is not differentiable, we use universal quantization (Choi et al., 2019), where every element of the latent space is shifted by the same random sample $u \sim \mathcal{U}\left(-\frac{\Delta}{2}\frac{\Delta}{2}\right)$. Regarding probability models, we use the fully factorized distribution by Ballé et al. (2018) for the base distribution $p_\theta\left(\mathbf{z}_0\right)$ while for the factor-out layers, we use a conditional distribution (see appendix for more details).

To get different quality levels on the rate-distortion curve, we train a single model using a fixed λ value, and at test time, we reach different rate-distortion points by varying the quantization step size used in the latent representation. To achieve best performance, we propose to fine-tune the quantization step values by minimizing the rate-distortion loss for various values of λ.

3 EXPERIMENTS

The model we consider in our experiments uses 8 additive layers at each level. Each of these coupling layers uses half of the channels. The transformation network $t(\cdot)$ is a ResNet with two blocks. In addition to the traditional image compression codecs, JPEG and BPG, we compare our model with recent learning based autoencoder models. In order to properly compare the performance, we train on the same dataset and refer to the retrained versions of Ballé et al. (2017), Ballé et al. (2018) and Minnen et al. (2018a) respectively as *AE + Fully-Factorized*, *AE + Hyperprior* and *AE + CM Hyperprior*. For completeness, we add the numbers published in the original paper when available. To better understand the limits on autoencoder based models, we also train a version of the networks to achieve the highest reconstruction without any rate penalty, referred to as *AE Limit*.

Image Compression - Fixed Resolution When using normalizing flows related work (Hoogeboom et al., 2019) only considered fixed size images. In the same spirit, our first experiment is based on a fixed size of 64×64. For this, we resized the images from ImageNet to a resolution of 64×64 as described in (Chrabaszcz et al., 2017).

Image compression results are presented in the supplementary material in Figure 6a. The autoencoder based models have a hard limit on the image reconstruction quality contrary to our solution that consistently increases in quality as more information is transferred. This limit is shown with a horizontal line and the value is obtained by training autoencoder networks to achieve the highest reconstruction without any rate penalty.

Image Compression - Arbitrary High Resolution The proposed architecture is fully convolutional and the probability distribution of the latents corresponding to the coarse level is modeled globally. As a result, the model can be trained on a fixed resolution and applied to arbitrary sized images. We use the COCO dataset (Lin et al., 2014) for training and extract image patches of size 128×128. We evaluate the trained models on the full images of the Kodak [1] and the Tecnick (Asuni & Giachetti, 2013) data sets. The results are visualized in Figure 3a and in the supplementary material in Figure 7.

In addition to this, the clear benefit of our normalizing flow based model is the capacity to target a large range of compression rates from low bit-rate to high quality results. This allows to better

[1] Downloaded from `http://r0k.us/graphics/kodak/`

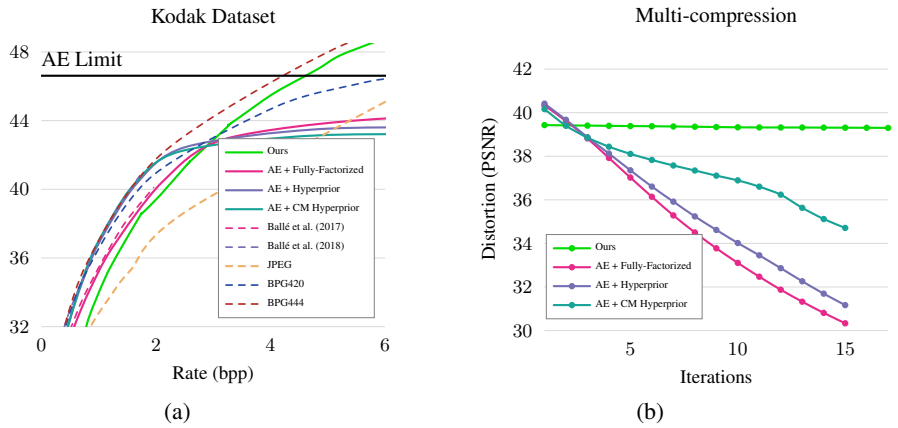


Figure 3: (a) Image compression results on high resolution images. Our model is not limited on the reconstruction quality contrary to auto-encoder models as indicated by the *AE Limit* line. (b) Distortion measurements after successive re-encodings.

match the behavior of traditional codecs and thus largely closes an important gap between learning based and traditional approaches.

Another interesting property is the model's possibility of progressively transmitting the image. By leveraging the hierarchical architecture the latents can be sent level by level and hence progressively improving the quality of the image (Figure 4 in Appendix).

Quasi Lossless Reencoding Besides the capacity to handle a larger range of quality levels than existing learned image compression techniques, our proposed method offers additional advantages. Since we use normalizing flows, there is a one-to-one mapping between quantized latent values and the corresponding reconstructed image. This means that compressing the decoded image again will result in the same latent values. Once the image quality is fixed, chaining multiple compression and decompression steps does not change the quality of the image and retains rate-distortion performance. This is in contrast to existing autoencoder based solutions that can quickly deteriorate and produce visible artifacts as content is reencoded several times (see Figure 1). This feature is important in video production pipelines where image and video content might be composited and edited by different parties requiring reencodings in the process. An evaluation is shown in Figure 3b, where both architectures, ours and autoencoder based approaches, start from the same image quality. All approaches achieve a constant bit-rate for all the steps, but the autoencoder reconstruction quality solution drops by 1db after only 2 steps and up to 10db after 15 steps.

Limitations In the low bit-rate part of the curve, autoencoder based solutions are the best performing. This is due to the more powerful transformations that are in place, in particular the use of nonlinear transformations (GDN layers). For the moment we only use additive coupling layers in our network and there is potential for substantial improvements as better bijective layers are developed for normalizing flows.

4 Conclusion

In this paper, we explored an approach for lossy image compression based on normalizing flows. To the best of our knowledge, we are the first to evaluate the potential of normalizing flows in the task of lossy compression. Furthermore, we have showcased a number beneficial properties inherent to our solution, such as the capacity to reencode images multiple times in a quasi lossless way closely retaining rate distortion performance. This has been problematic with autoencoder based methods. We believe the results obtained in this work are encouraging as there is still a lot of potential for improvement and would push new research works to explore new bijective layers, architecture designs and refined training procedures.

REFERENCES

Nicola Asuni and Andrea Giachetti. TESTIMAGES: A large data archive for display and algorithm testing. *J. Graphics Tools*, 17(4):113–125, 2013. doi: 10.1080/2165347X.2015.1024298. URL https://doi.org/10.1080/2165347X.2015.1024298.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *ICLR*, 2017.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *ICLR*, 2018.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 3146–3154. IEEE, 2019. doi: 10.1109/ICCV.2019.00324. URL https://doi.org/10.1109/ICCV.2019.00324.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *CoRR*, abs/1707.08819, 2017. URL http://arxiv.org/abs/1707.08819.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6421–6429, 2019.

Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. *CoRR*, abs/1905.07376, 2019. URL http://arxiv.org/abs/1905.07376.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 10236–10245, 2018. URL http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *CVPR*, 2019.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, 2018.

David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 10794–10803, 2018a.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*. 2018b.

Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *ICML*, 2017.

Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. *arXiv*, 2018.

Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Trans. Circuits Syst. Video Techn.*, 17(9):1103–1120, 2007. doi: 10.1109/TCSVT.2007.905532. URL https://doi.org/10.1109/TCSVT.2007.905532.

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. *ICLR*, 2016.

A APPENDIX

In this section we present an additional experiment on progressive transmission, describe the datasets and the details of the model's architecture and provide additional RD-curves. While some hyper-parameters differ between datasets, the common architecture is the same for all experiments. We further provide a pseudo code for the entropy coding used in our compression method.

A.1 DETAILS ABOUT PROBABILITY MODELS

To model the base distribution $p_\theta(\mathbf{z}_0)$ we use the fully factorized distribution by Ballé et al. (2018). For each channel dimension c, a neural network is used to learn the cumulative density function F_c. Considering probability independence between the latent elements $\hat{z}_0^{j,c}$, we have

$$p_\theta(\hat{\mathbf{z}}_0) = \prod_c \prod_j \left(F_c(\hat{z}_0^{j,c} + \frac{\Delta}{2}) - F_c(\hat{z}_0^{j,c} - \frac{\Delta}{2}) \right). \tag{5}$$

For the factor-out layers, we use a conditional distribution. Let $\hat{z}^j$ be an element in the latent encoding $\hat{\mathbf{z}}_l$. We model its probability with a discrete logistic distribution $\mathrm{DLogistic}(\hat{z}_j \mid \mu_j, \sigma_j)$ defined as:

$$\mathrm{DLogistic}(\hat{z}_j \mid \mu_j, \sigma_j) = \int_{\hat{z}_j - \frac{\Delta}{2}}^{\hat{z}_j + \frac{\Delta}{2}} \mathrm{Logistic}(z' \mid \mu_j, \sigma_j)\, dz'. \tag{6}$$

Parameters μ_j and σ_j at every position j are computed by a parameterized neural network $\phi(\mathbf{h}_l)$. Given this probability model, the sampling path amounts to using $\tilde{z}_j = \mu_j$.

A.2 ADDITIONAL EXPERIMENTS ON PROGRESSIVE TRANSMISSION

Scalable video coding (SVC) (Schwarz et al., 2007) allows to transmit only parts of a bit stream with the goal of retaining rate-distortion performance w.r.t. the partial stream that could represent a lower spatial resolution, a lower temporal resolution, or a lower fidelity content. As such SVC provides interesting opportunities for graceful degradation and flexible bit-rate adaptation in the context of video streaming. However, due to added complexity, it has not found widespread industry adoption. Aside from SVC, image standards such as JPEG and JPEG2000 also allow progressive transmission. This can be seen as a more fine grained way of scaling image fidelity.

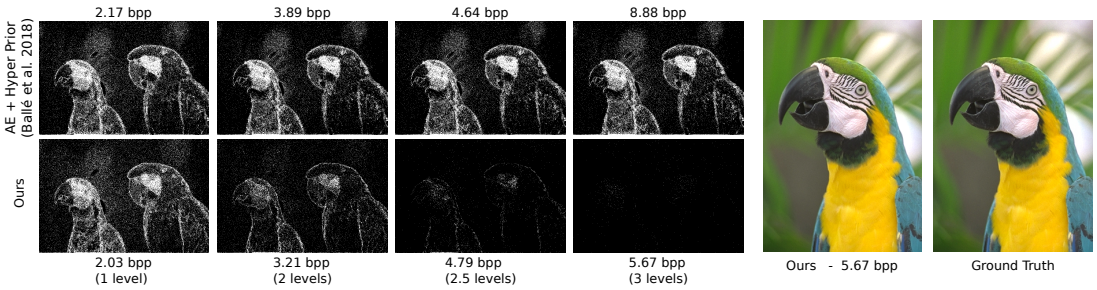


Figure 4: From lower bit-rate values to near lossless quality. On the left side we highlight every pixel with an error on the color value. As levels are encoded and the bit-rate increases, our solution reaches near lossless quality levels. Visual comparison with the ground truth is shown on the right side.

Our solution uses a hierarchical model with factor out distributions and thus is designed to inherently allow us to store details in the top layers and coarse information in the bottom layers. However, contrary to (Hoogeboom et al., 2019) these intermediate layers are optimized to have best rate-distortion performance. When only using the information stored in the bottom layer $\hat{\mathbf{z}}_0$, we can automatically set the values on the remaining layers to their estimated means without requiring to transmit any further information. In this way, we obtain a decoded image that has competitive rate-distortion performance w.r.t this partial stream of data. I.e. the results are comparable with those

produced by autoencoders in terms of rate *and* distortion (see Figure. 4, 1 level). To incrementally scale to a higher quality image, it is sufficient to only send the remaining latents $\hat{\mathbf{z}}_1$ (Figure. 4, 2 levels), or the full encodings $\hat{\mathbf{z}}_2$ (Figure. 4, 3 levels). It is also possible to partially send data and achieve progressive scaling, following a predefined pattern. This is illustrated for the latent $\hat{\mathbf{z}}_2$ (in Figure. 4, 2.5 levels) where only a fraction (*50%*) of the values are sent.

A.3 Training Details

We describe the model illustrated in Figure 3 in the main paper. Independent of the dataset, the models have $L = 3$ levels with K steps each. Each levels starts with a *Squeeze*-layer. A single step consists of a random permutation and an *additive transformation*. A ResNet (Figure 5) with B blocks and C hidden channels is used as transformer network $t(.)$ (see also Figure 2 in the main paper).

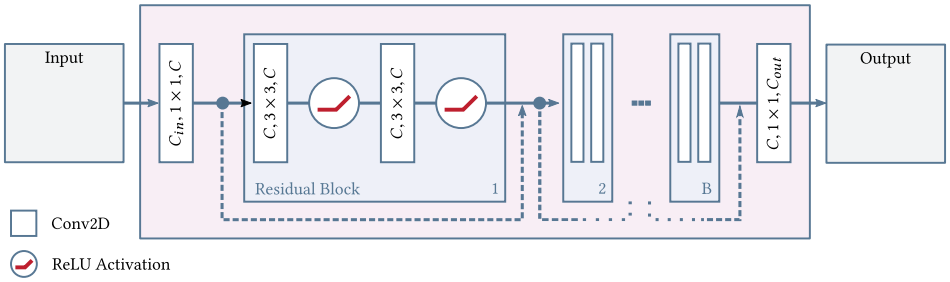


Figure 5: Overview of the architecture used as $t(.)$ and $\phi(.)$

Each factor-out layer divides the latents into two equal sized partitions, and only one of them is passed to the next level. We use a fully factorized model as prior $p(\mathbf{z}_0)$ and discrete logistic distributions as conditional distributions $p(\mathbf{z}_1|\mathbf{z}_0)$ and $p(\mathbf{z}_2|\mathbf{z}_1)$. The network $\phi(.)$ used to predict the parameters of the conditional distributions is also a ResNet with B blocks and C channels.

During training, we use the AdaMax optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3} (with $\epsilon = 10^{-7}$) and the quantization step is set to $\Delta = 1$.

After training, to optimize the quantization steps $\mathbf{\Delta}$ for a fixed $\lambda \in [1, 10^6]$ we use Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-1}.

A.4 Compression at different quality levels.

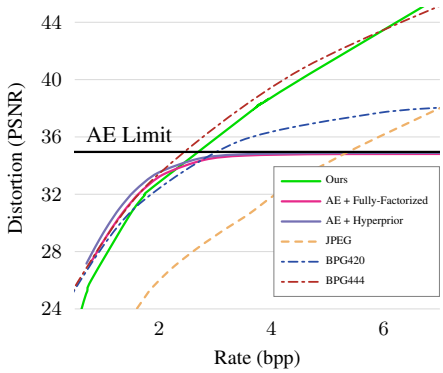
In order to avoid retraining image compression models for different rate-distortion levels Choi et al. (2019) use the Lagrange multiplier and the quantization bin size as conditioning variables to the network. It is unclear how such a strategy would fit in the normalizing flows model. Instead, to get different quality levels on the rate-distortion curve, we train a single model using a fixed λ value (see experiments for details), and at test time, we reach different rate-distortion points by varying the quantization step size used in the latent representation. In our case however, multiple levels are present and hence multiple step sizes must be set. To achieve best performance, we propose to fine-tune the quantization step values by minimizing the rate-distortion loss for various values of λ:

$$L(\mathbf{\Delta}; \mathbf{x}) = -\log p_\theta(\hat{\mathbf{z}}) + \lambda\, d(\mathbf{x}, \hat{\mathbf{x}}) \ , \tag{7}$$

where $\mathbf{\Delta}$ is the set of quantization steps to be estimated: a single scalar for each one of $\hat{\mathbf{z}}_2$ and $\hat{\mathbf{z}}_1$, and a distinct value for each channel of $\hat{\mathbf{z}}_0$.

A.4.1 ImageNet64

We resized the images from ImageNet to a resolution of 64×64 as described in (Chrabaszcz et al., 2017). For this dataset, the model is trained for 25 epochs, where each epoch consists of 40k batches of size 32. The results for the ImageNet64 dataset are visualized in Figure 6a.



(a) Results on ImageNet64. Our model is not limited on the reconstruction quality contrary to auto-encoder models, which have a hard limit indicated by the *AE Limit* line.

Parameter	**Value**
Levels L	3
Steps per level K	8
$\phi(.)$	ResNet ($B = 2$, $C = 64$)
$t(.)$	ResNet ($B = 2$, $C = 64$)
λ	500
batch size	32
learning rate	10^{-3}

(b) Model parameters used for ImageNet64 dataset

Figure 6: (a) Results for the ImageNet64 dataset. (b) Hyperparameters used for the model trained on the ImageNet64 Dataset

Baselines We used the same architecture proposed in (Ballé et al., 2018) with $M = 192$ and $N = 192$. We optimized for each model (*AE + Fully-Factorized*, *AE + Hyperprior*) the rate-distortion-loss for a fixed $\lambda = 0.3$ with Adam optimizer and learning rate $lr = 2 \cdot 10^{-4}$ for ~ 5.5 million iterations with batch size 32.

A.4.2 COCO

The COCO dataset consists of $\approx 285k$ images. We used the original train/validation/test - split. During training, we randomly cropped patches of size 128×128 due to the limitation of memory. The model is trained for 50 epochs.

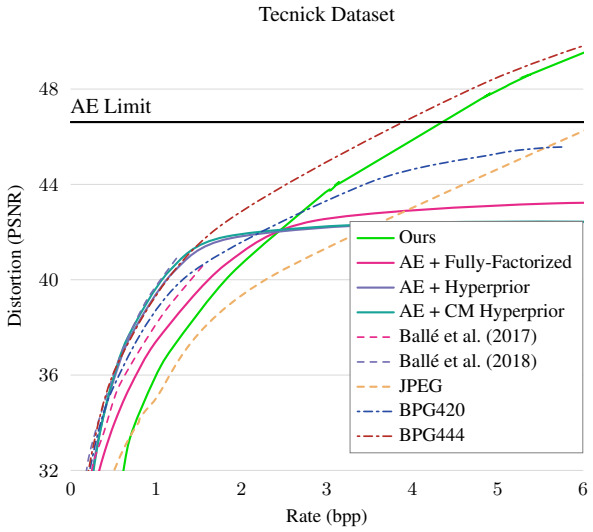


Figure 7: Results on COCO.

Baselines For all autoencoder based models (*AE-Limit*, *AE + Fully-Factorized*, *AE + Hyperprior* and *AE + CM Hyperprior*) we used the same architecture proposed in (Minnen et al., 2018a) with $M = 320$ and $N = 192$. We optimized for each model (*AE + Fully-Factorized*, *AE + Hyperprior*, *AE + CM Hyperprior*) the rate-distortion-loss for a fixed $\lambda = 0.15$ with Adam optimizer and learn-

Parameter	Value
Levels L	3
Steps per level K	8
$\phi\,(.)$	ResNet ($B = 3$, $C = 128$)
$t\,(.)$	ResNet ($B = 3$, $C = 128$)
λ	500
batch size	8
learning rate	10^{-3}

Figure 8: Hyperparameters used for the model trained on the COCO dataset

ing rate $lr = 2 \cdot 10^{-4}$ for ~ 1.2 Million iterations with batch size 32. During training, we randomly cropped patched of size 256×256.

A.5 Entropy Coding: Pseudo Algorithm

To further improve the bit-rate as the predictions reach higher probability values around the means (with respect to the quantization step), it becomes more beneficial to simply sample this value. In our compression scheme, we use a threshold on the probability value around the mean, $\mathrm{DLogistic}\,(\mu_j \,|\, \mu_j, \sigma_j) > P_{\text{thresh}}$, for which no information is transferred and the mean μ_j is used ($z_j = \mu_j$). In all our experiments $P_{\text{thresh}} = 0.9$.

The pseudo code in Algorithm 1 describes how our thresholding is considered in entropy encoding and decoding. The thresholding is performed based on the probability of the predictions (see end of Chapter 3 in the main paper). As the threshold can be arbitrarily set in principle, it offers another degree of freedom when realizing progressive transmission.

Algorithm 1: Pseudo code for entropy coding the latents

Function `entropy_encode` (*Latents* $[\hat{\mathbf{z}}_0^*, \hat{\mathbf{z}}_1^*, \hat{\mathbf{z}}_2^*]$) **:**
  $\mathbf{b}$ = `encode` $(\hat{\mathbf{z}}_0^*, P[\hat{\mathbf{z}}_0^*])$
  for $l = 1 \rightarrow 2$ **do**
    `/* for each position` j `in the latent of level` l `*/`
    foreach $z_j \in \hat{\mathbf{z}}_l^*$ **do**
      μ_j = `mean` $\left(P[z_j \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*]\right)$
      if $P[\mu_j \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*] \leq P_{thresh}$ **then**
        $\mathbf{b}$ += `encode` $\left(z_j, P[\hat{\mathbf{z}}_j^* \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*]\right)$
  return $\mathbf{b}$

Function `entropy_decode` (*bitstream* $\mathbf{b}$) **:**
  $\hat{\mathbf{z}}_0^*$ = `decode` $(\mathbf{b}, P[\hat{\mathbf{z}}_0^*])$
  for $l = 1 \rightarrow 2$ **do**
    `/* for each position` j `in the latent of level` l `*/`
    foreach $z_j \in \hat{\mathbf{z}}_l^*$ **do**
      μ_j = `mean` $\left(P[z_j \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*]\right)$
      if $P[\mu_j \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*] \leq P_{thresh}$ **then**
        z_j^* = `decode` $\left(\mathbf{b}, P[z_j \mid \hat{\mathbf{z}}_{l-1}^*, \ldots, \hat{\mathbf{z}}_0^*]\right)$
      else
        $z_j^* = \mu_j$
  return $[\hat{\mathbf{z}}_0^*, \hat{\mathbf{z}}_1^*, \hat{\mathbf{z}}_2^*]$