

---

# Lossy Image Compression with Normalizing Flows

---

Leonhard Helming<sup>1</sup> Abdelaziz Djelouah<sup>2</sup> Markus Gross<sup>1</sup> Christopher Schroers<sup>2</sup>

<sup>1</sup>Department of Computer Science  
ETH Zurich, Switzerland

<sup>2</sup>DisneyResearchStudios  
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 that have been established and refined over many decades. 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. Due to that, they inherently limit the range of quality levels that can be covered. In contrast, traditional approaches in image compression allow for a larger range of quality levels. Interestingly, they employ an invertible transformation before performing the quantization step which explicitly discards information. Inspired by this, we propose a deep image compression method that is 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. In addition to this, we demonstrate further advantages unique to our solution, such as the ability to maintain constant quality results through re-encoding, even when performed multiple times. To the best of our knowledge, this is the first work to explore the opportunities for leveraging normalizing flows for lossy image compression.

## 1 Introduction

Given the extremely large share of visual data in today's internet traffic, improvements in lossy image compression are likely to have an important impact. In the recent years, several works (Toderici et al., 2016; Rippel & Bourdev, 2017b; Mentzer et al., 2018; Minnen et al., 2018) 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).

Contrary to traditional image compression codecs that use handcrafted features, deep learning based methods are able to directly minimise the rate-distortion objective to obtain optimal nonlinear encoding and decoding transforms along with the probability models required for entropy coding. 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 (2017b) use a pyramidal encoder to extract features of various scales. Following these initial efforts, related works have focused on building better probability models Mentzer et al. (2019); Ballé et al. (2018); Minnen et al. (2018), addressing multiple quality levels Choi et al. (2019) or the very low bitrate regime Agustsson et al. (2019).

However, 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. As such, they cannot address high quality levels well. Using a recurrent

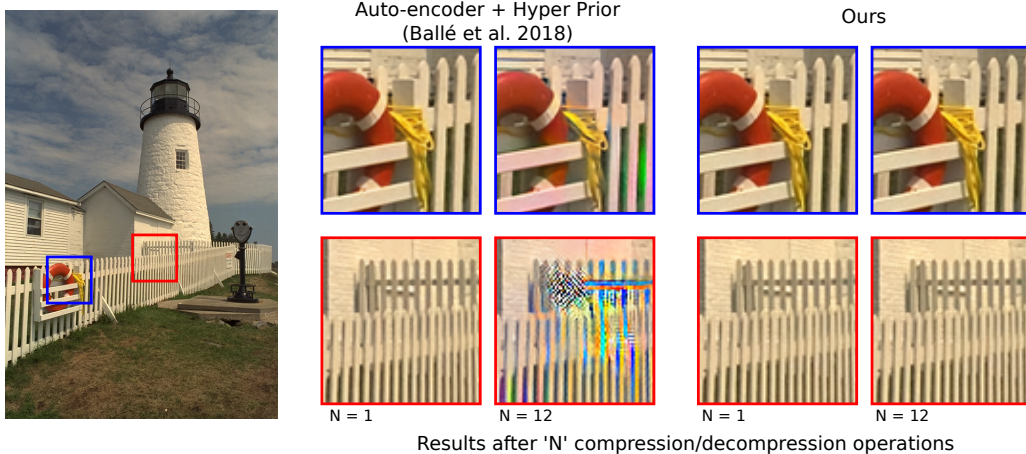


Figure 1: Using normalizing flows, image quality and bit-rate are not affected by multiple compression/decompression operations. With auto-encoder based solutions (Ballé et al., 2018), we notice color shifts and strong artifacts while the bit-rate increases.

architecture as proposed by Toderici et al. (2016) to iteratively achieve multiple compression rates can partially mitigate such problems, but this requires multiple passes and is not competitive in rate-distortion performance in general when compared to more recent learning based methods.

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. Interestingly, this strategy is related to traditional compression approaches that use lossless invertible transformations, such as the Discrete Cosine Transform (DCT), before a quantization step that explicitly discards information. 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. Figure 1 shows an example for this where multiple compression and decompression operations are run. The reconstruction quality of autoencoders quickly drops and noticeable artifacts appear in the image, while the reconstruction quality as well as the bit rate remain constant for our model. This property can be particularly interesting for media production pipelines where images are sent between multiple entities and may be compressed multiple times. Another beneficial property of our approach is progressive reconstruction, i.e. the ability to gradually increase reconstruction quality as more information is sent. Even though we do not yet outperform existing autoencoder based approaches over the full range of bit rates, we believe the results obtained in this work are very encouraging nevertheless, especially as this marks a first step with a lot of potential for future work.

Summing up, the contribution of this paper is three fold: (1) To the best of our knowledge, our work is the first to explore the potential of normalizing flows for lossy image compression. (2) Our model achieves the widest range of quality levels among learned image compression methods and can outperform existing autoencoder solutions in the high quality regime. (3) We demonstrate additional advantages to our solution such as the capacity to maintain rate-distortion performance during multiple reencodings, as well as the ability to perform progressive reconstruction.

Before explaining our method in detail in Chapter 3, we cover the most important background on normalizing flow in Chapter 2. We then set our method into context with related work (Chapter 4) and show a comprehensive experimental evaluation (Chapter 5) before finally concluding (Chapter 6).

## 2 Background on Normalizing Flow

In normalizing flow (Rezende & Mohamed, 2015) the objective is to map an arbitrary distribution to a base distribution, which is done through a change of variable. Let’s consider two random variables  $X$  and  $Z$  that are related through the invertible transformation  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d, f(\mathbf{z}) = \mathbf{x}$ . In this case, the distributions of the two variables are related through

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) |\det(J_f(\mathbf{z}))|^{-1}, \quad (1)$$

where  $J_f(\mathbf{z})$  is the Jacobian matrix of  $f$ . In normalizing flow, a series  $f_K, \dots, f_1$  of such mappings is applied to transform a simple probability distribution into a more complex multi-modal distribution:

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \prod_{k=1}^K \left| \det \left( \frac{\partial \mathbf{h}_{k-1}}{\partial \mathbf{h}_k} \right) \right|^{-1}, \quad (2)$$

where  $\mathbf{x} = (f_K \circ \dots \circ f_1)(\mathbf{z})$  is the bijective relationship between  $\mathbf{z}$  and  $\mathbf{x}$ , and  $\mathbf{h}_k = f_k(\mathbf{h}_{k-1})$  the intermediate outputs with  $\mathbf{h}_K \triangleq \mathbf{x}$  and  $\mathbf{h}_0 \triangleq \mathbf{z}$ .

Recent works (Kingma & Dhariwal, 2018; Dinh et al., 2017) have shown the great potential of using normalizing flow as a generative model. We can consider an image as a high-dimensional random vector  $\mathbf{x}$  that has an unknown distribution  $p(\mathbf{x})$ . In flow based generative models the observation  $\mathbf{x}$  is generated from a latent representation  $\mathbf{z}$ :

$$\mathbf{x} = f_\theta(\mathbf{z}) \quad \text{with} \quad \mathbf{z} \sim p_\theta(\mathbf{z}), \quad (3)$$

where  $f_\theta$  is a sequence of bijective transformations and  $p_\theta(\mathbf{z})$  a tractable distribution with  $\theta$  being the parameters of the model. Following equation 2, we can learn the parameterized distribution  $p_\theta(\mathbf{x})$  from a discrete set of  $N$  images, by minimizing the following negative log-likelihood (*nll*) objective

$$\mathcal{L}_{nll} \simeq \frac{1}{N} \sum_{i=1}^N -\log p_\theta(\bar{\mathbf{x}}^{(i)}). \quad (4)$$

Here  $\bar{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} + \xi$  with  $\xi \sim \mathcal{U}(0, a)$  and  $a$  is determined by the discretization level of the data. Next we present the bijective mappings used in our model.

**Coupling layers** split the input into two partitions, where one conditions a neural network to modify the remaining channels. The *additive* coupling layer is the one used as building block for our models (Figure. 2). Given an input tensor  $\mathbf{u}$  with partitions  $(\mathbf{u}_a, \mathbf{u}_b)$ , the additive layer is defined as the following mapping

$$f(\mathbf{u}) = (\mathbf{u}_a, \mathbf{u}_a + t(\mathbf{u}_b)), \quad (5)$$

where  $t$  is modeled as a neural network. The partitioning is random and defined during the initialization of the model. The inverse is easy to compute and the determinant is equal to 1.

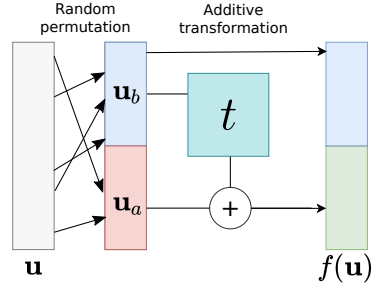


Figure 2: Additive coupling layer

**Factor-out layers** allow a coarse to fine modeling and a simplification of the representation by only further processing a part of the input features. The concept was proposed in the *RealNVP* (Dinh et al., 2017). Formally this is expressed as

$$(\mathbf{z}_1, \mathbf{h}_1) = g_1(\mathbf{x}) \quad \text{and} \quad \mathbf{z}_0 = g_0(\mathbf{h}_1), \quad (6)$$

where the input vector  $\mathbf{x}$  is first mapped to  $(\mathbf{z}_1, \mathbf{h}_1)$ , then only a part of the latents,  $\mathbf{h}_1$ , are further processed and mapped to  $\mathbf{z}_0$ . The resulting latent representation for  $\mathbf{x}$  is  $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_0)$ . In addition to computation efficiency, this defines a conditional dependency between the latents.

### 3 Lossy Image Compression with Normalizing Flows

We propose using the normalizing flows for lossy image compression, and in particular we design an architecture based on the bijective layers described in the previous section. The key properties are the coarse-to-fine representation obtained with the multi-level factor-out layers and the usage of series of simple additive couplings. The model is illustrated in Figure 3 and consists of 3 different levels bijectively mapping any input image  $\mathbf{x}$  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 Equation 4. Adapted to our model this equation becomes

$$\mathcal{L}_{nll} = -\frac{1}{N} \sum_{i=1}^N \log p_\theta(\mathbf{z}_0) + \log p_\theta(\mathbf{z}_1|\mathbf{z}_0) + \log p_\theta(\mathbf{z}_2|\mathbf{z}_1) \quad (7)$$

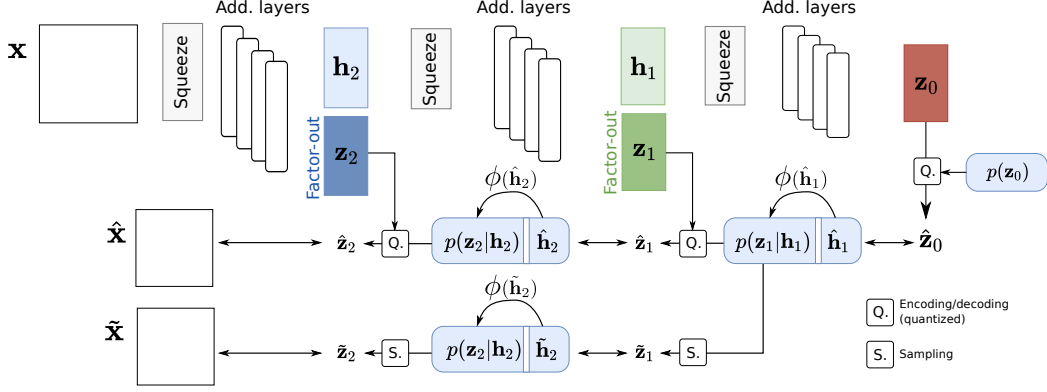


Figure 3: Overview of the flow based image compression architecture. The input image  $\mathbf{x}$  is processed through a 3 level network, where each level consists of a squeeze operation followed by a series of additive layers before a factor-out. The input image  $\mathbf{x}$  is mapped to its latent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

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  $\hat{\mathbf{z}}_l = r(\mathbf{z}_l)$

$$\hat{\mathbf{x}} = f_\theta(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2), \quad (8)$$

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

$$\mathcal{L}_{rdl} = -\log p_\theta(\hat{\mathbf{z}}) + \lambda d(\mathbf{x}, \hat{\mathbf{x}}) \quad (9)$$

with  $d(\mathbf{x}, \hat{\mathbf{x}})$  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 hierarchical architecture and the properties of the factor out distributions of our model. This is achieved by only transmitting a part of the latent space and sampling the missing values. This is illustrated in Figure 3, 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:

$$\mathcal{L}(\theta) = -\log p_\theta(\hat{\mathbf{z}}) + \lambda (d(\mathbf{x}, \hat{\mathbf{x}}) + d(\mathbf{x}, \tilde{\mathbf{x}})) \quad (10)$$

**Quantization.** Since the rounding operation is a step function, it is not differentiable and an approximation is necessary to allow gradient based optimization. In this work, we employ universal quantization proposed by Choi et al. (2019) to relax this problem. Universal quantization shifts every element of the latents by a common random sample  $u \sim \mathcal{U}(-\frac{\Delta}{2}, \frac{\Delta}{2})$ :

$$\hat{\mathbf{z}} = \text{round}(\mathbf{z} + \mathbf{u}) - \mathbf{u}, \quad \mathbf{u} = [u, u, \dots, u]. \quad (11)$$

This allows the model to learn correlation of the elements in the latent space. For the round(.) operation, we use the straight-through estimator (Bengio et al., 2013) and set the gradients to the identity,  $\nabla_{\mathbf{x}} \text{round}(\mathbf{x}) = \mathbf{I}$ .

**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

$$p_\theta(\hat{\mathbf{z}}_0) = \prod_c \prod_j \left( F_c(\hat{z}_0^{j,c} + \frac{\Delta}{2}) - F_c(\hat{z}_0^{j,c} - \frac{\Delta}{2}) \right). \quad (12)$$



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  $\text{DLogistic}(\hat{z}_j | \mu_j, \sigma_j)$  defined as:

$$\text{DLogistic}(\hat{z}_j | \mu_j, \sigma_j) = \int_{\hat{z}_j - \frac{\Delta}{2}}^{\hat{z}_j + \frac{\Delta}{2}} \text{Logistic}(z' | \mu_j, \sigma_j) dz'. \quad (13)$$

Parameters  $\mu_j$  and  $\sigma_j$  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  $\tilde{z}_j = \mu_j$ .

**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  $\lambda$  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  $\lambda$ :

$$L(\mathbf{x}; \Delta) = -\log p_\theta(\hat{\mathbf{z}}) + \lambda d(\mathbf{x}, \hat{\mathbf{x}}), \quad (14)$$

where  $\Delta$  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  $\hat{\mathbf{z}}_0$ .

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,  $\text{DLogistic}(\mu_j | \mu_j, \sigma_j) > P_{\text{thresh}}$ , for which no information is transferred and the mean  $\mu_j$  is used ( $z_j = \mu_j$ ). In all our experiments  $P_{\text{thresh}} = 0.9$ .

## 4 Related Work

Recent works made significant progress in applying neural networks to image compression. Toderici et al. (2016, 2017) first introduced a general framework for variable-rate image compression using an LSTM based architecture. Advantageously, the method directly outputs a binary representation and has no explicit limit on the amount of information that can be encoded, however their empirical results suggest this solution does not necessarily lead to rate-distortion optimality.

Better performances have been achieved using autoencoders to learn in an end-to-end fashion an optimal transformation to a latent representation, as well as the probability distribution required for entropy coding to the final bitstream (Ballé et al., 2017; Theis et al., 2017; Rippel & Bourdev, 2017a). For example Rippel & Bourdev (2017a) use a pyramidal encoder to extract features of various scales. After quantizing the encoding, they use adaptive arithmetic coding and adaptive code length regularization to reduce the length of the bitstream. Ballé et al. (2017) propose a model that uses newly proposed nonlinear transformations (GDN) which implement a form of local gain control. To further improve performance, more focus was placed on the entropy model of the learned latent representation. In particular, Ballé et al. (2018) use side information in the form of a hyperprior to condition the Gaussian mixture model used to entropy code the latents. In a different direction, Mentzer et al. (2018) use a 3D-CNN to learn a conditional probability model of the latent distribution. State of the art results were achieved by combining these strategies Minnen et al. (2018).

At this stage we find it interesting to draw back attention to the traditional approaches in image compression. These methods use an invertible transformation before a quantization step where information is explicitly discarded. This is the case for JPEG (Wallace, 1991) where image patches are transformed with a discrete cosine transformation (DCT) then the obtained coefficients are scaled, quantized and entropy encoded to retrieve the bit stream. This strategy ensures that it is always possible to increase the quality by sending more data. Recent image codecs have improved on some particular aspects, like JPEG 2000 (Taubman & Marcellin, 2002) which replaces the DCT with a wavelet-based method or BPG (Fab, 2015) and Webp (Goo, 2010) that explored directions such as intra prediction and in-loop filtering, but all are capable of leveraging more data for better quality and even provide a lossless mode.

Some learned lossless compression approaches have been proposed recently: Mentzer et al. (2019) use a hierarchical probabilistic model; Hoogeboom et al. (2019) introduced integer coupling layers to learn a discrete mapping from the image space to a discrete base distribution which can be used for lossless image compression. We note however that given their probability model, the solution is limited to a fixed size and encoding a large image requires patch-wise processing. This work is the first to explore using normalizing flows for *lossy* image compression and demonstrating compression results on a wide range of bit-rates.

## 5 Experiments

The model we consider in our experiments uses 8 additive layers at each level. Each of these coupling layers (Figure 2), uses half of the channels. The transformation network  $t(\cdot)$  is a ResNet with two blocks. 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  $\Delta = 1$ . In addition to the traditional image compression codecs, JPEG and BPG, we compare our model with recent state of the art learning based models. In order to properly compare the performance, we train on the same dataset (COCO with  $\approx 285k$  images) and refer to the retrained versions of (Ballé et al., 2018) and (Ballé et al., 2017) as *AE + Hyperprior* and *AE + Fully-Factorized* respectively. Furthermore, we always train a single model and use different quantization steps at test time to create the rate-distortion curves.

For completeness, we add the numbers published in the original paper where available. In addition to this, and 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. This is done by dropping the rate penalty term during training. Models trained this way are referred to as *AE Limit*.

### 5.1 Image Compression - Fixed Resolution

When using normalizing flows related work both in image synthesis (Kingma & Dhariwal, 2018; Dinh et al., 2017) and recently in image compression (Hoogeboom et al., 2019) only considered fixed size images. In the same spirit, and to demonstrate the potential of normalizing flows, our first experiment is based on a fixed size of  $64 \times 64$ . For this, we resized the images from ImageNet to a resolution of  $64 \times 64$  as described in (Chrabaszcz et al., 2017). In this case, the models is trained for 25 epochs, where each epoch consists of 40k batches of size 32.

Image compression results are presented in Figure 4. 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.

In the low bit-rate part of curve, auto-encoder 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.

### 5.2 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 (see Equation 12). 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  $128 \times 128$ . We evaluate the trained models

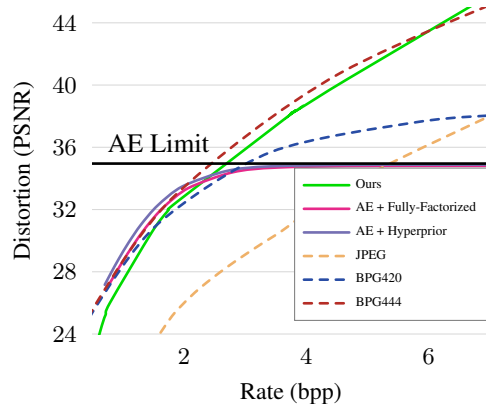


Figure 4: Results on ImageNet64. Our model is not limited on the reconstruction quality contrary to auto-encoder models, which have a hard limit indicated by the *AE Limit* line.

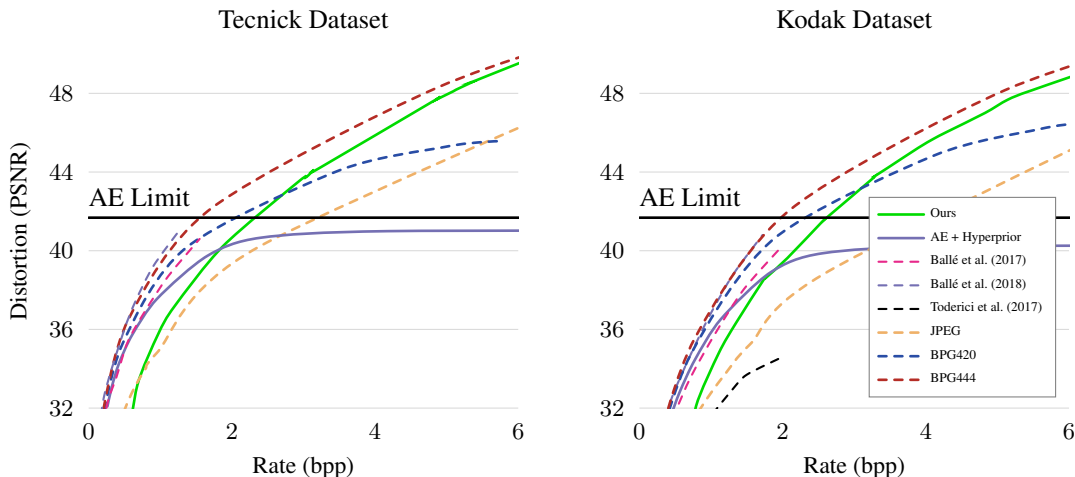


Figure 5: Image compression results on high resolution images. Our model is not limited on the reconstruction quality contrary to auto-encoder models as indicated by the *AE Limit* line.

on the full images of the Kodak <sup>1</sup> and the Tecnick (Asuni & Giachetti, 2013) data sets. The results are visualized in Figure 5.

For completeness, we also show the PSNR values available from the author Ballé et al. (2018) for the Kodak dataset. Here only a smaller range of bitrates is covered as multiple individual models were trained. Furthermore, the training was relying on a significantly larger dataset. As the PSNR values are higher than our retrained *AE + Hyperprior* model, this may indicate that further gains for our flow based model might also be possible when training on a larger dataset.

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

### 5.3 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). Being able to properly reencode can be important in video production pipelines where the image and video content might be composited and edited by different parties requiring reencodings in the process. An evaluation is shown in Figure 6, where both our model and the autoencoder based approach of Ballé et al. (2018) start from the same image quality. Both approaches achieve a bit-rate of 1.7bpp for

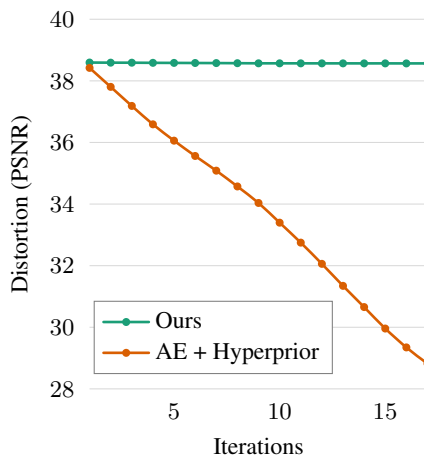


Figure 6: Successive re-encodings

<sup>1</sup>Downloaded from <http://r0k.us/graphics/kodak/>

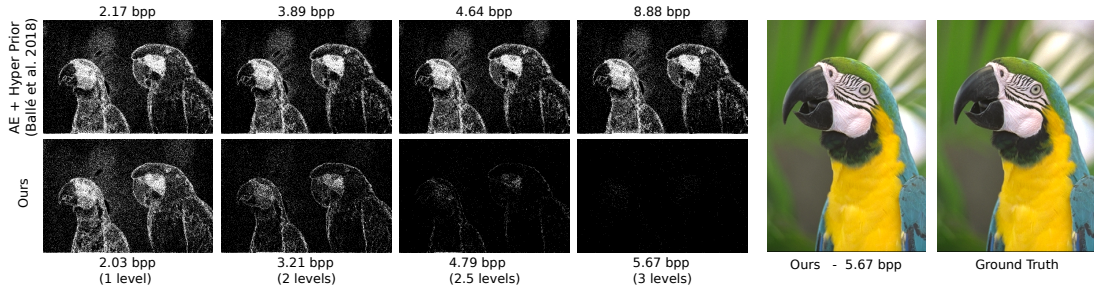


Figure 7: From lower bit-rate values to near lossless quality. On the left side we highlight every pixel with an error on the color value. As levels are encoded and the bit-rate increases, our solution reaches near lossless quality levels. Visual comparison with the ground truth is shown on the right side.

all the steps, but the autoencoder reconstruction quality solution drops by almost 1db after only 1 step and up to 10db after 15 steps.

#### 5.4 Towards Scalable Coding and 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.

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 (Hoogetboom et al., 2019) these intermediate layers are optimized to have best rate-distortion performance. When only using the information stored in the bottom layer  $\hat{z}_0$ , 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. 7, 1 level). To incrementally scale to a higher quality image, it is sufficient to only send the remaining latents  $\hat{z}_1$  (Figure. 7, 2 levels), or the full encodings  $\hat{z}_2$  (Figure. 7, 3 levels). It is also possible to partially send data and achieve progressive scaling, following a predefined pattern. This is illustrated for the latent  $\hat{z}_2$  (in Figure. 7, 2.5 levels) where only a fraction (50%) of the values are sent.

## 6 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: 1) We are able to cover the widest range of quality levels with a single trained model ranging from low bitrates to almost lossless quality thereby bridging an important gap that had remained between currently used autoencoder based methods and traditional transform codecs. 2) We show that we can reencode images multiple times in a quasi lossless way closely retaining rate distortion performance. This has been problematic with autoencoder based methods. 3) We describe different ways of achieving scalable coding or progressive reconstruction with our model.

These properties make normalizing flows a very exciting candidate for lossy image compression models even though we do not yet outperform existing autoencoder based approaches over the full range of bit rates. 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

- Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., and Gool, L. V. Generative adversarial networks for extreme learned image compression. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
- Asuni, N. and Giachetti, A. 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>.
- Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *ICLR*, 2017.
- Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. *ICLR*, 2018.
- Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL <http://arxiv.org/abs/1308.3432>.
- Choi, Y., El-Khamy, M., and Lee, J. 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>.
- Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *CoRR*, abs/1707.08819, 2017. URL <http://arxiv.org/abs/1707.08819>.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL <https://openreview.net/forum?id=HkpbnH91x>.
- Djelouah, A., Campos, J., Schaub-Meyer, S., and Schroers, C. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6421–6429, 2019.
- BPG Specification*. Fabrice Bellard, 2015. version 0.9.5.
- Webp*. Google, 2010.
- Hoogeboom, E., Peters, J. W. T., van den Berg, R., and Welling, M. Integer discrete flows and lossless compression. *CoRR*, abs/1905.07376, 2019. URL <http://arxiv.org/abs/1905.07376>.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (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>.
- Kingma, D. P. and Dhariwal, P. 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>.
- Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL <http://arxiv.org/abs/1405.0312>.
- Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z. Dvc: An end-to-end deep video compression framework. In *CVPR*, 2019.
- Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. Conditional probability models for deep image compression. In *CVPR*, 2018.

- Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Gool, L. V. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10629–10638, 2019.
- Minnen, D., Ballé, J., and Toderici, G. D. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*. 2018.
- Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1530–1538, 2015. URL <http://proceedings.mlr.press/v37/rezende15.html>.
- Rippel, O. and Bourdev, L. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2922–2930. JMLR. org, 2017a.
- Rippel, O. and Bourdev, L. Real-time adaptive image compression. In *ICML*, 2017b.
- Rippel, O., Nair, S., Lew, C., Branson, S., Anderson, A. G., and Bourdev, L. Learned video compression. *arXiv*, 2018.
- Schwarz, H., Marpe, D., and Wiegand, T. 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>.
- Taubman, D. S. and Marcellin, M. W. Jpeg2000: Standard for interactive imaging. *Proceedings of the IEEE*, 90(8), 2002.
- Theis, L., Shi, W., Cunningham, A., and Huszár, F. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.
- Toderici, G., O’Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R. Variable rate image compression with recurrent neural networks. *ICLR*, 2016.
- Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., and Covell, M. Full resolution image compression with recurrent neural networks. In *CVPR*, 2017.
- Wallace, G. K. The JPEG still picture compression standard. *Commun. ACM*, 34(4):30–44, 1991. doi: 10.1145/103085.103089. URL <https://doi.org/10.1145/103085.103089>.



## A Supplementary Material: Lossy Image Compression with Normalizing Flows

In this section we describe the datasets and the details of the model architecture. While some hyperparameters 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.

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 8) with  $B$  blocks and  $C$  hidden channels is used as transformer network  $t(\cdot)$  (see also Figure 2 in the main paper).

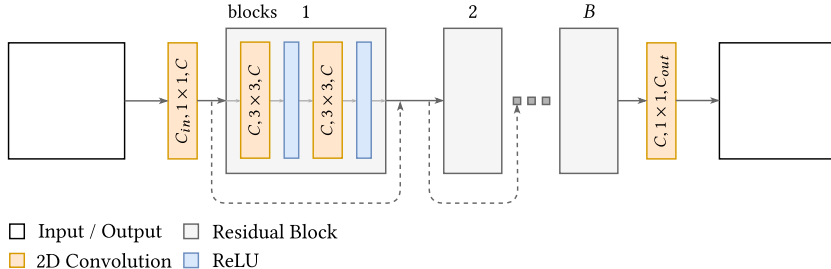


Figure 8: Overview of the architecture used as  $t(\cdot)$  and  $\phi(\cdot)$

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  $\phi(\cdot)$  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  $\Delta$  for a fixed  $\lambda \in [1, 10^6]$  we use Adam optimizer (Kingma & Ba, 2015) with learning rate  $10^{-1}$ .

### A.1 ImageNet64

We resized the images from ImageNet to a resolution of  $64 \times 64$  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.

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

Figure 9: Model parameters used for 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* and *AE + Hyperprior*) the rate-distortion-loss for a fixed  $\lambda = 0.3$  with Adam optimizer and learning rate  $lr = 2 \cdot 10^{-4}$  for  $\sim 5.5$  million iterations with batch size 32.

## A.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  $128 \times 128$  due to the limitation of memory. The model is trained for 50 epochs.

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

Figure 10: Model parameters used for COCO 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* and *AE + Hyperprior*) the rate-distortion-loss for a fixed  $\lambda = 0.15$  with Adam optimizer and learning rate  $lr = 2 \cdot 10^{-4}$  for  $\sim 1.2$  Million iterations with batch size 32. During training, we randomly cropped patched of size  $256 \times 256$ .

## A.3 Entropy Coding: Pseudo Algorithm

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

---

```

1 Function entropy_encode(Latents  $[\hat{z}_0^*, \hat{z}_1^*, \hat{z}_2^*]$ ):
2   b = encode( $\hat{z}_0^*, P[\hat{z}_0^*]$ )
3   for  $l = 1 \rightarrow 2$  do
4     /* for each position  $j$  in the latent of level  $l$  */
5     foreach  $z_j \in \hat{z}_l^*$  do
6        $\mu_j = \text{mean}(P[z_j | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*])$ 
7       if  $P[\mu_j | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*] \leq P_{\text{thresh}}$  then
8         b += encode( $z_j, P[\hat{z}_j^* | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*]$ )
9   return b
10 Function entropy_decode(bitstream b):
11    $\hat{z}_0^* = \text{decode}(\mathbf{b}, P[\hat{z}_0^*])$ 
12   for  $l = 1 \rightarrow 2$  do
13     /* for each position  $j$  in the latent of level  $l$  */
14     foreach  $z_j \in \hat{z}_l^*$  do
15        $\mu_j = \text{mean}(P[z_j | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*])$ 
16       if  $P[\mu_j | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*] \leq P_{\text{thresh}}$  then
17          $z_j^* = \text{decode}(\mathbf{b}, P[z_j | \hat{z}_{l-1}^*, \dots, \hat{z}_0^*])$ 
18       else
19          $z_j^* = \mu_j$ 
20   return  $[\hat{z}_0^*, \hat{z}_1^*, \hat{z}_2^*]$ 

```

---