Lossy Image Compression with a Compressive Autoencoder

Alfredo Reichlin - alfrei@kth.se
Luca Marson - lmarson@kth.se

KTH Royal Institute of Technology

Abstract. In this project we analyze the problem of image compression by means of a deep neural network. We propose a model composed by two parallel networks, a convolutional autoencoder and a 3d convolutional network. The latter is used to learns a conditional probability distribution of the latent representation of the first autoencoder. The main challenge of this model is being able to find a balance in the trade-off between the distortion rate of the reconstructed image and the bitrate of the latent representation.

Keywords: image compression, CNN, context model, tensorflow

1 Introduction / Problem formulation

The task of lossy image compression consists in minimizing the amount of bits needed for storing and trasmission by trading quality and bitrate. Standard lossy compression algorithms such as JPEG rely on heuristics handcrafted by experts to capture and exploit images features to achieve satisfactory compression rates. This domain is one of the many that in the near future could see some improvements thanks to the growing research interest in training Deep Neural Networks. DNNs could in fact be used to detect dependencies among regions of images by learning them from image datasets. The growth in amount of computational power and bandwidth available to the public could allow the diffusion of new and more flexible compression techniques adaptable to specific tasks such as video streaming or virtual reality.
For all these reasons we decided to read trough the state of the art research works in lossy image compression with DNN and chose [1] to investigate what are the implications of training a deep network for this task. In particular, we selected a subset of ImageNet and performed some experiments to test the proposed architecture and measure the results.

2 Related works

Image compression with the use of deep neural networks has become an active area of research in the last few years. Toderici et al. [2] explored the use of different kind of Recurrent Neural Network based encoders and decoders for full

resolution lossy image compression. They claim to be the first to be able to outperform standard compression methods such as JPEG on the Kodak dataset [3]. In [4] the authors used RNNs to tackle the problem of thumbnail compression beyond the capabilities of existing codecs.
Another approach to lossy image compression is the use of generative models to reconstruct missing information. In [5] a Generative Adversarial Network is trained to perform extreme image compression at very low bitrates. The authors propose an interesting approach consisting in recognizing and synthesizing unimportant regions of the image such as streets or trees, thus requiring the storage of significantly less information with respect to standard compression techniques.
Compressive autoencoders have been widely deployed for image compression tasks. Input images are transformed into bitstreams which can be further compressed using lossless compression algorithms such as Huffman coding or adaptive arithmetic coding [1] [2]. The method presented in [1] is built on the model proposed by Theis et al. [6], but differs in the use of a very deep convolutional autoencoder instead of a RNN one. Their approach takes into consideration entropy coding to capture the distribution of the bit streams and reduce the coding rates as in [2], but they achieve it through a context model that has to be trained concurrently with the autoencoder.
One issue of lossy image compression with DNN is the choice of an appropriate measure of quality, as mentioned in [2]. The often used Mean Square Error may not in fact be sufficient for this task because it does not take into consideration global features of an image. The Multiscale Structural Similarity [7] is instead a widely used metrics [2][1] since it takes into consideration how human perceive the quality of an image.

3 Approach

Lossy image compression consists in minimizing the distortion rate of the reconstructed image as well as minimizing the bitrate of the compressed representation. In general when the bitrate decreases the distortion increases and vice versa posing the challenge of finding a balance between the two when designing a compression algorithm. Keeping this in mind we used the same approach of [1] where they modeled the loss function as the combination of two parameters: an index to measure the difference between the input image and the decompressed one and a measure for the bitrate. By including both these two factors in the loss the network is forced to find a middle ground between the two. In particular we used two different indexes for the distortion rate, the standard mean squared error and the MultiScale-SSIM [7] which is supposed to reflect better how humans perceive differences between images. For what concerns the bitrate the standard quantification function used is the entropy of compressed image. The resulting loss can be written as:

$$d\left(\mathbf{x}, \hat{\mathbf{x}}\right) + \beta H\left(\hat{\mathbf{z}}\right) \tag{1}$$

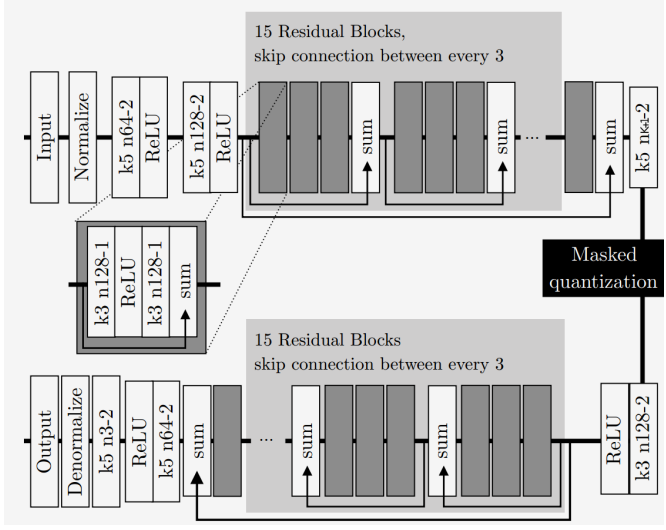


Fig. 1: Architecture of the autoencoder designed by [1]. We modifed the original network by setting the number of output channels of the last layer of the encoder to K + 1 to account for y.
"k5 n128-2" refers to a convolutional layer with kernel size 5, 128 output channels and stride 2.

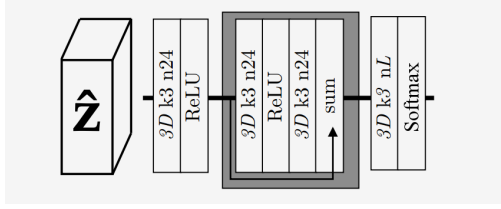


Fig. 2: Architecture of the context model designed by [1]. We modified the original network by setting the number of output channels of the last layer to L (number of centroids) and by adding a softmax layer, in order to obtain a probability distribution of the latent representation $\hat{z}$ as output

Where d represent a generic distortion function while H is the entropy. Here $\mathbf{x}$, $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are respectively the input image, the reconstructed image and the compressed image.

While the distortion rate can be easily computed using classic formulations, the entropy poses some challenges. In fact, this measure requires a probability distribution for each quantized symbol used for the compressed latent representation of the input image. Following [1] approach we used a parallel model that takes the latent representation found by the encoder and outputs a context model of the image by learning a convolutional probabilistic model of it.

The overall model consists of a very deep convolutional autoencoder, a shallow context model and a quantizer. The output of the encoder is further compressed using an importance map and quantized so that it can be losslessly encoded using adaptive arithmetic encoding or similar approaches.

3.1 Context Model

The context model is used to determine the entropy of the compressed image. The distribution of the probability that each value corresponds to every symbol $p(\hat{\mathbf{z}})$ can be factorized as the product of conditional distributions as in PixelRNN [8]:

$$p(\hat{\mathbf{z}}) = \prod_{i=1}^{m} p(\hat{z}_i \mid \hat{z}_{1:i-1}) \tag{2}$$

The context model can be used to approximate this distribution:

$$P_{i,l}(\hat{\mathbf{z}}) \approx p(\hat{z}_i = c_l \mid \hat{z}_{1:i-1}) \tag{3}$$

To impose this conditional constrain among the values of $\hat{z}$ a mask can be applied to the filters of the convolution layers of the network as stated in [1], however these masks have to be adapted to 3D convolutional layers. The model is then trained to maximize the likelihood of the assigned centroids for each value of the latent representation of the compressed image. This can be achieved by minimizing the cross entropy between the real and the approximated probability distributions that [1] prove to be an estimate of the entropy of the latent representation. At the same time the context model has to preserve a low distortion rate between the reconstructed image and the original one. The resulting loss function can be written as:

$$l_{CM} = \frac{1}{B} \sum_{j=1}^{B} \left[d(\mathbf{x}^{(j)}, \hat{\mathbf{x}}^{(j)}) + \beta \sum_{i=1}^{m} -log(P_{i,I(\hat{z}_i)}) \right] \tag{4}$$

where B is the batch size, β is a scalar value and $P_{i,I(\hat{z}_i)}$ is the probability of each value of the latent representation to belong to the assigned centroid.

Training the context model improves the estimate of the entropy of $\hat{\mathbf{z}}$ which is used by the autoencoder to explore the rate-distortion trade-off. This approach relies heavily on the two networks being trained in parallel so as not to make one of the two to prevail. As [1] states this method is similar to how Generative Adversarial Networks work with the difference that the two networks are cooperative and not competitive.

3.2 Autoencoder

The main component of the model is a deep convolutional autoencoder used to compress and restore the input image. The encoder part of this network outputs a latent representation z with a reduced height and width but a considerably

larger depth with respect to the input set to K. These channels are used to encode information at different spatial locations of the image. However, often images have a non uniform arrangement of details and a fixed number of channels per spatial location can result in a considerable sub-optimal use of data. To adapt the number of channels used by the network, following the idea of [1], we make the encoder generate also a second output y used to build a tridimensional importance map to force the encoder to use a variable amount of bits per location. The map is expanded from y as follows:

$$m_{i,j,k} = \begin{cases} 1, & \text{if } k < y_{i,j} \\ (y_{i,j} - k), & \text{if } k \leq y_{i,j} \leq k+1 \\ 0, & \text{if } k+1 > y_{i,j} \end{cases} \tag{5}$$

z is then multiplied pointwise to the ceiling of m. The ceil operation is considered to be an identity function during the backward pass due to its not differentiable nature.

Quantization Quantization consists of assigning each value of the latent representation to a specific symbol, in our case we used 6 centroids. The resulting image can thus be processed using adaptive arithmetic encoding. However the quantization process is not differentiable and cannot be used directly in the backward pass. To avoid this issue we relied on the method deployed by [1], where they use standard quantization for the forward pass:

$$\hat{z}_i = Q\left(z_i\right) = argmin_j \|z_i - c_j\| \tag{6}$$

and a differentiable soft quantization in the backward pass:

$$\tilde{z}_i = \sum_{j=1}^{L} \frac{exp\left(-\sigma \|z_i - c_j\|\right)}{\sum_{l=1}^{L} exp\left(-\sigma \|z_i - c_l\|\right)} c_j \tag{7}$$

The decoder, then, restores the original image starting from the quantized and masked latent representation.

Loss function By including the importance map m only in the loss function of the autoencoder and not in the context model the network needs to learn how to represent $\hat{\mathbf{z}}$ in a meaningful way. Following the same reasoning for the entropy of the context model we can add m to the loss which results in the function:

$$l_{AE} = \frac{1}{B} \sum_{j=1}^{B} \left[d(\mathbf{x}^{(j)}, \hat{\mathbf{x}}^{(j)}) + \beta \sum_{i=1}^{m} -\lceil m_i \rceil log(P_{i,I(\hat{z}_i)}) \right] \tag{8}$$

4 Results

The architecture for the deep convolutional autoencoder and the context model are showed respectively in Figure 1 and Figure 2.
To asses quantitatively the performances of the network we computed the distortion rate as: $d(\mathbf{x}, \hat{\mathbf{x}}) = 100 * (1 - \text{MS-SSIM}(\mathbf{x}, \hat{\mathbf{x}}))$. For the compression size we considered the bit per pixel unit.

We trained the network on a subset of 87500 images taken from the ImageNet dataset from the Large Scale Visual Recognition Challenge 2012 [9]. We pre-processed the data by equally selecting images from all the 1000 classes and by performing random crops of size 160x160. To test the network performances we used the Kodak dataset [3]. Since the training dataset is quite big, we implemented the network using TensorFlow 1.8 to maximize the performances and to visualize statistics through tensorboard and ran the code on a Nvidia Tesla K80. Even with this configuration the training phase took around 18h for each configuration we tried.

For all our experiments we used the Adam optimizer with mini-batch size of 30 images and we ran the models for 6 epochs. For the smoothing function of the quantization formula (7) we used $\sigma = 10$ which is higher than the original implementation of [1] in order to have a closer representation to the standard quantization formula. Moreover we clipped the entropy to a minimum value of t = 1e-3. To quantize the latent representation of the image $\hat{\mathbf{z}}$ we used L = 6 centroids. For all the experiments we ran we set the number of channels of the latent representation K to 32. All the resulting compressed images had thus a maximum of 1.29 bits per pixels assuming that each quantized symbol requires $log_2(L) = 2.59$.

The hardest part of the training phase turned out to be finding a good balance between the distortion term and the bitrate term in the loss function of the networks, thus finding a proper value for β. We started off by setting $\beta = 10$ as suggested by [1], but this resulted into a sudden increase of the loss value due to the network tendency to put more weight on the entropy term instead of reaching a balance for the trade-off. To avoid this problem we decreased the value of β such that the entropy value was in the same range of values such as for the distortion rate.

The parameter which most affected the perforamces of the network is the learning rate. We noticed that with a learning rate of 0.0001 the MS-SSIM decreased faster with respect to a learning rate of 0.00001, but the training phase was much more unstable. Figure 3 shows that with a high learning rate the network diverged after within the first epoch.

Keeping this in mind, we decided to start with a high learning rate of 0.0001 and to decrease it every 100 updates hoping to achieve a more stable training. We also decreased the regularization term of each layer by 10. Figure 4 show that we achieve a much lower distortion rate with this setting, but the network was not able to reach the 6th epoch.

We then tried to reduce the depth of the autoencoder to see if the learning phase could be more stable and if the performaces decreased. Figure 5 shows

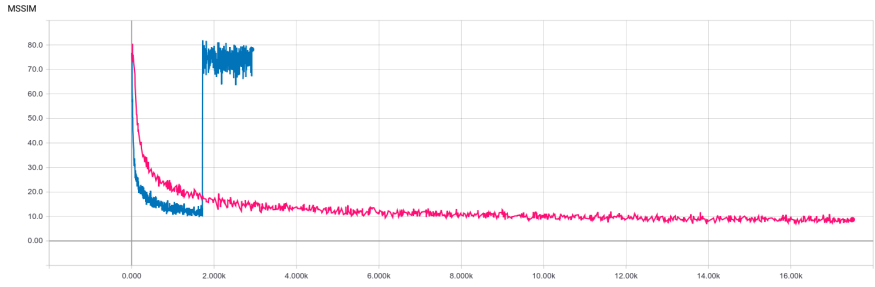


Fig. 3: Comparison of the distortion rate with different learning rate.

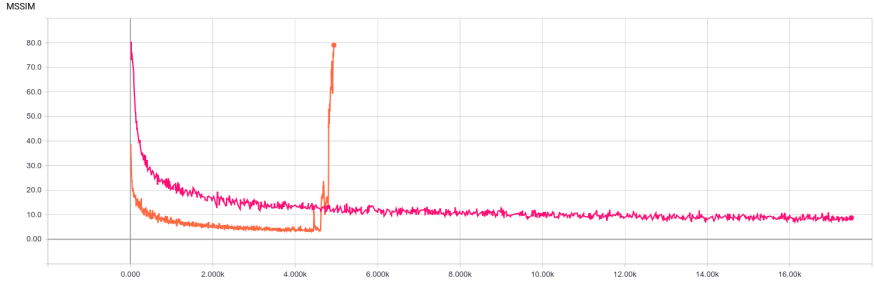


Fig. 4: Comparison of the distortion rate: deacaying learning rate (orange curve) vs fixed learning rate

how in the short run the value of the accuracy is not affected in a considerable way when reducing the number of layers from 70 to 34. However this behaviour can be explained by the fact that our network was not trained enough to let it campture high resolution details and thus such an higher number of layers is probably not needed in general.

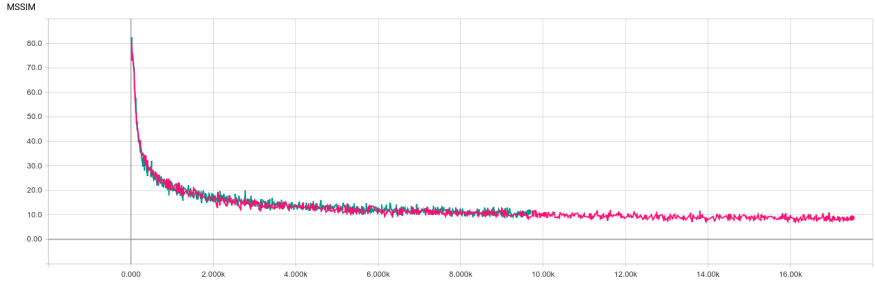
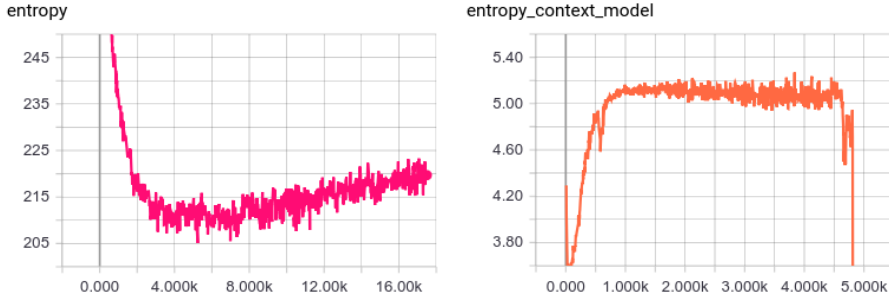


Fig. 5: Comparison of the distortion rate: deep model (pink curve) vs more shallow architecture

For what concerns the entropy, when the networks converges to a valid solution the general behaviour is for it to decrease in the beginning of the training while increasing slowly when the model starts to favour the distortion rate over the entropy. However when the model diverges to a high loss solution the entropy suddenly drops to zero because the autoencoders starts to output uniform images.



entropy of a converging model entropy of a divergent model

Fig. 7

We tested the two best models on the Kodak dataset. One deep and one more shallow. Even if the shallow model had an overall better accuracy with respect to the deep one, colors are less saturated.

Original Kodak 16 image Original Kodak 21 image

Fig. 9

Reconstructed Kodak 16 image with the shallow model, MS-SSIM is 0.9369

Reconstructed Kodak 21 image with the shallow model, MS-SSIM is 0.9357

Fig. 11

Reconstructed Kodak 16 image with the deep model, MS-SSIM is 0.8545

Reconstructed Kodak 21 image with the deep model, MS-SSIM is 0.895

Fig. 13

5 Conclusions

The overall architecture of the network is not particularly complex based on a convolutional autoencoder and a shallow context model. However training this network proved to be not an easy task. First of all the depth of the network requires very big datasets and a considerable computational power. Moreover the loss function and the parallel nature of the two network required a very careful analysis and choice of the parameters. However, despite the complexity of this approach, the model has the advantage of being able to minimize the distortion-entropy trade-off with the continuos training of a single architecture. Furthermore, the use of the context model in conjuction with the importance map has the benefit of adapting the dimensions of the compressed image based on the number of details of the original image in such a way to optimize the number of bits required to store such a representation.

Despite the network being able to achieve very interesting results, we think that improving the context model with more powerful networks such as PixelRNN [8] or PixelCNN [10] could improve the entropy estimation. In addition, we think that generative networks such as variational autoencoders or generative adversarial networks could be used to restore the lost information from the reconstructed image to decrease the loss term.

References

1. Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., Gool, L.V.: Conditional probability models for deep image compression. CoRR **abs/1801.04260** (2018)
2. Toderici, G., Vincent, D., Johnston, N., Hwang, S.J., Minnen, D., Shor, J., Covell, M.: Full resolution image compression with recurrent neural networks. CoRR **abs/1608.05148** (2016)
3. : Kodak photocd dataset. http://r0k.us/graphics/kodak/
4. Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M., Sukthankar, R.: Variable rate image compression with recurrent neural networks. CoRR **abs/1511.06085** (2015)
5. Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., Gool, L.V.: Generative adversarial networks for extreme learned image compression. CoRR **abs/1804.02958** (2018)
6. Theis, L., Shi, W., Cunningham, A., Huszár, F.: Lossy image compression with compressive autoencoders. CoRR **abs/1703.00395** (2017)
7. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. Signals, Systems and Computers **Conference Record of the Thirty-Seventh Asilomar Conference on, volume 2, pages 13981402. Ieee, 2003**
8. van den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. CoRR **abs/1601.06759** (2016)
9. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. CoRR **abs/1409.0575** (2014)
10. van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K.: Conditional image generation with pixelcnn decoders. CoRR **abs/1606.05328** (2016)