Autoencoders with Variable Sized Latent Vector for Image Compression

Alekh Karkada Ashok
RV College of Engineering,
Bangalore, India
alekhka@gmail.com

Nagaraju Palani
RV College of Engineering,
Bangalore, India
nagarajup@rvce.edu.in

Abstract

Learning to compress images is an interesting and challenging task. Autoencoders have long been used to compress images into a code of small but fixed size. As different images need different sized code based on their complexity, we propose an autoencoder architecture with a variable sized latent vector. We propose an attention based model which attends over the image and summarizes it into a small code. This summarization is repeated many times depending on the complexity of the image, producing a new code each time to encode new information so as to get a better reconstruction. These small codes then form sub-units of the final code. Our approach is quality progressive and has flexible quality setting which are desirable properties in compression. We show that the proposed model shows better performance compared to JPEG.

1. Introduction

Image compression is an important area of research. With ever increasing media consumption, it is the need of the hour to find more efficient compression methods. With changing media needs it is also time to do away with hard-coded compression schemes.

Deep learning has revolutionized image recognition and realistic image generation. Image compression can also be improved with deep learning techniques as compression relies heavily on pattern recognition. Compression involves identifying the structure/pattern present and coming up with a representation which exploits the identified redundancy.

As deep learning methods exhibit exemplary feature extraction performance, compression techniques based on them have shown superior performance over traditional hard coded codecs. Toderici et al. [13],[14] show usage of recurrent neural networks for compression and demonstrate superior performance against JPEG, JPEG2000 and WebP. Johnston et al. [7] employ a loss weighted with SSIM to outperform BPG, WebP, JPEG2000, and JPEG. Theis et al. [12] use compressive autoencoders and Ballé et al.[3] use a modified form of quantization. Rippel and Bourdev [10] employ pyramidal analysis and codelength regularization to outpeform JPEG, JPEG2000, BPG and WebP by a significant margin.

Autoencoders are a popular architecture for image compression [12], [3], [15], [1], [9]. Autoencoders are made of three distinct parts - the encoder, the latent vector and the decoder. The encoder encodes the input into a latent vector of a fixed size and the decoder learns to reconstruct the original input from the latent vector. For compression applications, the latent vector has lesser dimensions compared to the dimensions of the input. In effect, the encoder has to learn to discover structures in the input and exploit the redundancy to be able to encode into a space of lesser dimension. On the other hand, the decoder has to learn to understand the encoding and reconstruct the original input from the compressed representation.

An acute drawback of autoencoders is the fixed size of the latent vector. This forces all images to be represented by a code of fixed size. This is not ideal because the codelength should depend on the complexity of the image. A busy image with lot of objects might need a longer codeword than a simple image. This makes variable sized latent vector a necessity. Variable sized latent vector is not trivially achievable in normal autoencoders because of the nature of convolution and transposed convolution operations.

We propose an attention based model for this task. Attention mechanism has been successfully applied to machine translation [4],[2], image captioning [16], one shot image recognition [11] etc. Gregor et al. introduced a zoom-able and differentiable attention mechanism in DRAW [6] model. Many works from the past [5],[8] have shown that visual structure can be captured better by a sequence of glimpses as opposed to a single feed forward input.

2. Methods

The proposed architecture makes the following modifications to autoencoders: 1. attention mechanism replacing the feed-forward convolution pipeline in the encoder 2. se-

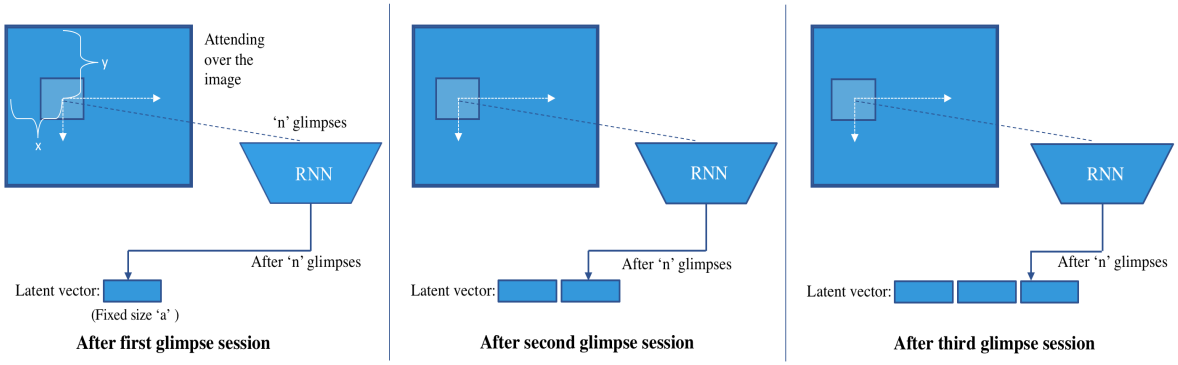

Figure 1. Three glimpse sessions of the encoder. After 'n' glimpses the hidden state of the RNN controller is taken as a codeword. Therefore, the codeword has the size of the hidden state of the RNN. Each newly produced codeword is appended to the latent vector array. The glimpse parameter generator neural network is omitted for simplicity.

ries of small codewords replacing the fixed-size latent vector 3. transposed convolution pipeline taking each codeword from the latent vector array and updating the reconstructed image

2.1. Encoder

The encoder summarizes the input image by attending over the image. It consists of a recurrent neural network controller which governs the area of the image to attend to. The attention mechanism takes in the image and glimpse parameters to produce a glimpse output.

$$G_t = attend(I, \Omega_t) \quad \text{where} \quad \Omega_t = W_g h_{t-1}$$

attend() is the attention mechanism and the glimpse parameters specify the location and size of the attention window. The glimpse parameters are obtained from the previous RNN hidden state using a small fully-connected neural network called as the glimpse parameter generator. The glimpse and the hidden state are used to produce the RNN's next hidden state. This encodes the image according to what is known from the current glimpse and what was previously known about the image. The next glimpse is dependent on the glimpse parameters which are in turn dependent on the hidden state of the RNN controller. In effect, the hidden state of the RNN holds what is already known about the image and dictates what new information should be brought in.

The attention mechanism used is identical to the one used by Shyam et al. [11] which is an improvement of the mechanism used by Gregor et al. [6] As cropping images is non-differentiable, soft attention is used where pixels are weighted according their 'importance'. This is done by placing an N x N grid of kernels over the image. In this work, Cauchy kernels have been used. The location of the grid (x,y) and its size (δ) is given by the glimpse parameters. The grid of kernels is placed on the image, with the central Cauchy kernel being located at (x,y). The kernels are placed with a stride of δ, which gives the spacing between the kernels. The δ in effect determines how large the attention window is. The parameters x,y,δ are obtained from the output ($\hat{x}$,$\hat{y}$,$\hat{\delta}$) of the glimpse parameter generator neural network by:

$$x = (S-1)\,\frac{(\hat{x}+1)}{2} \qquad y = (S-1)\,\frac{(\hat{y}+1)}{2}$$

$$\delta = \frac{S}{N}(1-|\hat{\delta}|) \qquad \gamma = e^{1-2|\hat{\delta}|}$$

The location of a Cauchy kernel at the i^{th} row, j^{th} column in terms of the pixel coordinates of the image is given by:

$\mu_X^i = x+(i-(N+1)/2)\delta \quad \mu_Y^j = y+(j-(N+1)/2)\delta$

The glimpse is calculated using separate horizontal and vertical filterbank matrices. These matrices are given by:

$$F_X[i,a] = \frac{1}{Z_X}\left\{\pi\gamma\left[1+\left(\frac{a-\mu_X^i}{\gamma}\right)^2\right]\right\}^{-1}$$

$$F_Y[j,b] = \frac{1}{Z_Y}\left\{\pi\gamma\left[1+\left(\frac{b-\mu_Y^j}{\gamma}\right)^2\right]\right\}^{-1}$$

Z_X and Z_Y are normalization constants such that they make $\Sigma_a F_X[i,a] = 1$ and $\Sigma_b F_X[j,b] = 1$

The glimpse output is given by:

$$attend(I, \Omega_t) = F_Y I F_X^T$$

The image is attended 'n' times to produce one codeword of size 'a' which is the size of the hidden state of the RNN. This constitutes one glimpse session. The state of the RNN after 'n' glimpses is taken as the codeword 'c'. This is repeated to produce further codewords as shown in Figure 1. After each codeword is produced, it is passed to the decoder which reconstructs the image from it, which is then compared to the original input image and the error

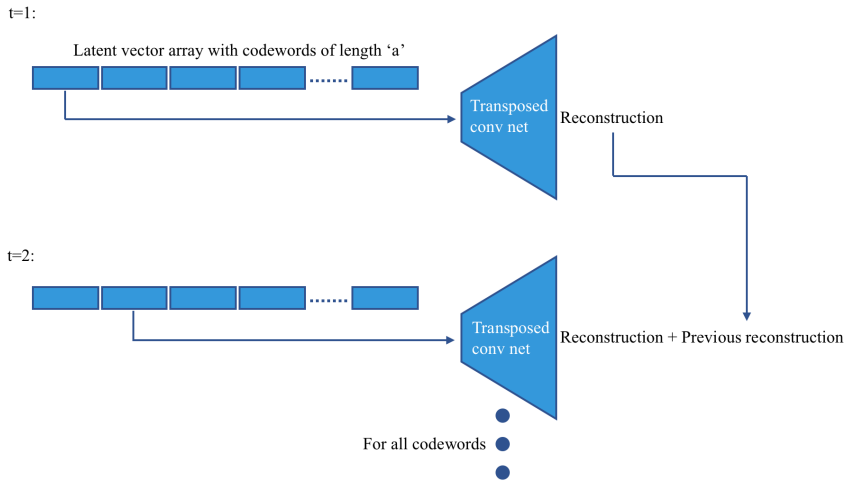

Figure 2. Steps for decoding two codewords of the latent vector array. The output from each codeword is used as an update or 'correction' to the reconstructed image.

is found out. With each codeword, the reconstructed image improves and the encoding is repeated until the reconstructed image reaches a desirable quality level. Since the size of the glimpse window is fixed, the glimpse obtained gets 'pixelated' when zoomed out (as stride (δ) is large), which gives an approximate about what the image holds and where in the image. This will further influence where the encoder should attend next.

2.2. Latent vector

Latent vector of fixed size in the autoencoder is replaced with an array 'L' of codewords, each having size 'a'. Each codeword 'c' produced by the encoder after each glimpse session is appended to the latent vector array. L_t denotes the latent vector after t glimpse sessions.

$$L_t = [L_{t-1}, c] \quad \text{where } c \text{ is the new codeword after 'n' glimpses}$$

2.3. Decoder

The decoder starts with an empty canvas and each codeword is used to update the canvas. Each codeword is input into the transposed convolutional network to generate a 2D array of size equal to the input image. This serves as the update:

$$reconstructed_t = reconstructed_{t-1} + W_{upconv}(L[t])$$

W_{upconv} represents the transposed convolutional network. Each output of the network is due to a new codeword from the latent vector array. Hence, each output is an update to the reconstructed image in view of the new information encoded in the codeword. Figure 2 shows two steps of the decoder, showing how the reconstruction is improved.

3. Experiments

3.1. Data

We trained the model on the MNIST dataset and the CLIC dataset provided by the Computer Vision Lab of ETH Zurich. The encoder and decoder learn to efficiently encode and decode the images in the dataset. Hence, dataset selection is crucial. A well rounded dataset covering all kinds of images required by the application is necessary for good performance.

3.2. Results and discussion

As outlined in Table 1, encoding and decoding is fast in Nvidia Geforce 920M, a lower end notebook GPU.

	Average time
Encoder	42ms
Decoder	32ms

Table 1. Average time taken for encoding and decoding on 920M GPU.

We observe that the approach produces good reconstructions even at low bits per pixel values. Figure 3 shows a comparison between our approach and JPEG. We see that in the case of JPEG, there is a drastic decrease in the quality and the presence of lot of artifacts. In our approach, artifacts are low and there is a good reconstruction even at low bits per pixel values. Figure 3 shows the image improving with additional codewords. Each codeword encodes more information about the image and thus the image improves with more codewords.

We see that our approach has several properties desirable in compression- 1. Quality progressive - the decoder

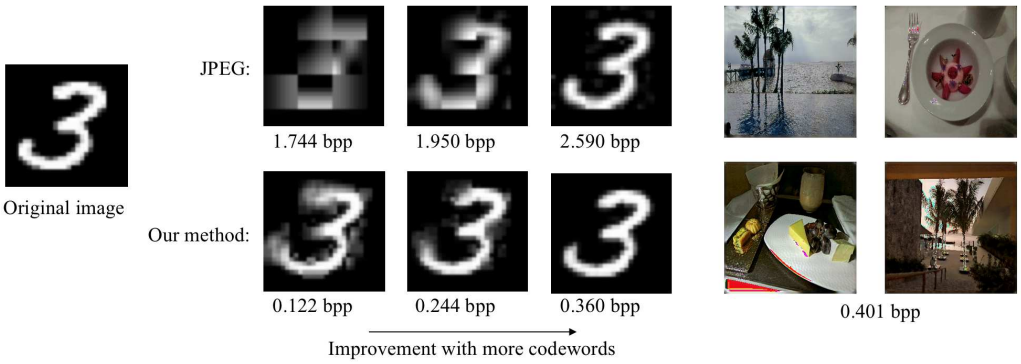

Figure 3. Reconstructed images for various bpp values. JPEG images of comparable quality are shown to take lot more bits per pixel. The reconstructions in our approach show a clear improvement in quality and reduction in artifacts as more codewords are used.

doesn't have to wait for the whole bitstream to arrive. The decoding can start with the first codeword itself and the image can be improved with subsequent codewords. 2. Quality flexible - The encoder can take in a "quality" parameter and encode until that quality is achieved. 3. Domain adaptable - Can be trained on a dataset of specific variety and obtain highly efficient compression for that domain.

4. Conclusion

The paper presented a novel modification to autoencoder architecture to make it more suitable for image compression. We showed that the approach gives good quality reconstructions even at low bpp values. We further showed that the approach has many desirable advantages.

Acknowledgement: We would like to thank Pranav Shyam, Vishwesh Nayak and Vignesh R for their extensive support and feedback while developing ideas and manuscript for this work.

References

[1] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *CoRR*, abs/1109.3737, 2011.

[6] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[7] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. *arXiv preprint arXiv:1703.10114*, 2017.

[8] H. Larochelle and G. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pages 1243–1251, USA, 2010. Curran Associates Inc.

[9] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. *arXiv preprint arXiv:1703.10553*, 2017.

[10] O. Rippel and L. Bourdev. Real-time adaptive image compression. *arXiv preprint arXiv:1705.05823*, 2017.

[11] P. Shyam, S. Gupta, and A. Dukkipati. Attentive recurrent comparators. *arXiv preprint arXiv:1703.00767*, 2017.

[12] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[13] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

[14] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. *arXiv preprint*, 2016.

[15] R. Torfason, F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Towards image understanding from deep compression without decoding. *arXiv preprint arXiv:1803.06131*, 2018.

[16] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.