CS230

https://github.com/decotoj/dnn-compression

Deep Neural Networks for Image Compression

Brian Graham
Stanford University
bgraham6@stanford.edu

Jake Decoto
Stanford University
decotoj@stanford.edu

Abstract

Image compression has historically relied upon approaches which either adjust the encoding of digital images (lossless compression) or adjust digital images in the frequency domain (lossy compression). While these methods are generally effective, recent research has focused on machine-learning based techniques to supplement, or replace, these approaches. We explore three existing state of the art deep learning algorithms from literature and present several extensions and modifications. Additionally, we capitalize on inherent advantages of different approaches and create a new hybrid algorithm.

1 Introduction

With the rise in streaming content, and increasing ease at which content may be generated, substantial reductions in bandwidth and storage may be achieved through even modest improvements in compression. However, complications may arise when implementing these reductions, for example efforts to improve compression may cause unacceptable reductions in visual quality. Indeed, one of the primary difficulties in designing an image compression algorithm is balancing rate (e.g., bitrate) with a measure of distortion (e.g., the so-called 'rate-distortion tradeoff').

Neural networks may provide clear advantages to the above-described difficulty through the automated learning of parameters which balance this tradeoff. The algorithms we describe below use respective loss functions in which a measure of rate is combined with a measure of distortion. While the algorithms attempt a similar balancing act, and use similar techniques to determine the distortion term, they take unique perspectives on how to define the rate-term of their loss functions.

The first algorithm, Balle [7], estimates rate using a Gaussian mixture model. Specifically, Balle describes using an analysis transform to map image pixels to a quantized latent variable representation. The rate term of Balle's loss function measures cross-entropy between the marginal distribution of the latent variables and a learned entropy model. The second algorithm, Mentzer [11], learns to reduce rate through a CNN-learned 'importance map'. The importance map, which was first described in Li [10], learns to allocate bitrate according to information content in local areas of an image . Mentzer discards more complicated entropy estimation techniques in favor of this learned importance map.

In this paper, we propose a combined Balle-Li algorithm which adjusts Balle to use the importance map concept outlined in Li. This novel algorithm achieved almost identical image quality as baseline Balle but with 12% better compression ratio, see Table 1.

2 Related work

Due to the relative infancy of neural-network based image compression techniques, and as cited herein, there are many disparate implementations. A literature review reveals that various neural network-based approaches are being used, such as CNNs [1] [7] [10] [11], RNNs [2], GANs[8], and Neural Networks to aid Haar Wavelet Compression methods [4]. We have focused on Balle [7], Li [10], and Mentzer [11] because of their impressive performance and the public availability of the Mentzer [11] and Balle [7] code.

3 Dataset and Features

Two data sets are used for training. On the authors' local machines, a small dataset - the Berkeley Segmentation Dataset, BSDS500 [5] - is used for rapid prototyping. This small 500 image data set is split 80/10/10 between a train, development, and test set. For final evaluation of promising algorithms training is performed on Amazon Web Services with a 20,000 image subset of the Google Open Images dataset. This dataset also includes bounding box labels for different classified features in the images.

4 Methods

4.1 Balle Algorithm

The Balle algorithm for image compression [9] [7] uses a network consisting of several convolution layers with non-linear activation functions. We started with, and then refactored and modified, a publicly available Tensorflow implementation. The baseline loss function, J, for this algorithm is identified below in equation 1 where MSE refers to Mean Squared Error between the input vector of pixel values x and output pixel values $\tilde{x}$ of an autoencoder. Rate refers to entropy of a latent variable representation of an input image. In Balle, the rate is determined based on (1) a likelihood associated with a Gaussian distribution of each latent variable and (2) a number of pixels. The hyperparameter λ augments the rate-distortion tradeoff.

$$J = \lambda * MSE + Rate \tag{1}$$

$$MSE = 1/m * \sum_{i=0}^{m} x_i - \tilde{x}_i \tag{2}$$

$$Rate = 1/m * \sum_{i=0}^{m} log(likelihoods)/(log(numpixels)) \tag{3}$$

Minor modifications to this algorithm included exploring different learning rates, filters per layer, batch size, and varying values of λ . More extensive modifications included rewriting the CNN based encoder and decoder to achieve varying compression rates and varying levels of learnable parameters with different activation functions. Only the most promising are presented here.

4.2 New Balle-Li Combined Algorithm

This paper introduces a new algorithm which combines the modified Balle algorithm with a written from scratch importance map network inspired by Li [10]. Figure 1 shows the retained elements from Balle (blue) along with the newly added elements (green).

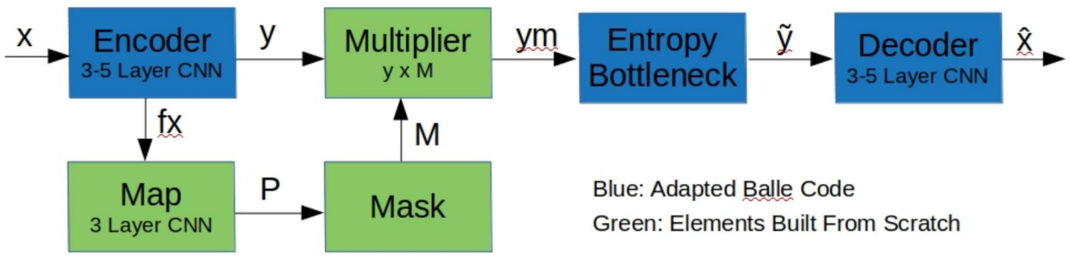

Figure 1: Network Structure for Balle (Blue) and Combined Balle/Li (Green) Algorithms

The basic innovation borrowed from Li is to use the output of an intermediate layer of a CNN encoder, fx, as an input to an importance map network. The importance map network is a multi-layer CNN with a sigmoid function for its final output. The sigmoid output, P, of the importance map network is then input to a Mask function which turns all values below a MASK threshold hyperparameter, nominally 0.5, to zero and all those above to 1.0. Because the gradient of the MASK function is 0 everywhere except at the boundary a customized function is substituted for the default tensorflow backpropagation behavior, following the methods described in recent work on binarized neural networks [12]. The binary output of the Mask function, M, is then multiplied by the output 'y' of the encoder and pushed through the rest of the Balle style network.

The net effect of the above-described Balle-Li algorithm is to enable the importance map network to learn a feature based function to ignore areas of the convolved image which can be reconstructed from neighboring values. Advantageously, the entropy bottleneck functionality from Balle is utilized to encode the entire convolved image space. Li [10], in contrast, describes that only portions of the mask multiplied output are encoded. Li then uses this as a substitute for the entropy encoding utilized in the Balle algorithm.

4.3 Mentzer Algorithm

As described above, a main insight to be extracted from Li is their utilization of an importance map as a substitute for entropy in an image. Mentzer [11] integrates this importance map concept into an end-to-end learning scheme. Specifically, Mentzer describes a loss function which combines distortion loss between an input image and its decoded form and a coding-rate loss. The rate loss is referred to as a masked-coding cost of a latent variable version of the input image. The masked-coding cost, which uses the learned importance map, is defined as:

$$MC(\tilde{z}_i) = \sum_{i=0}^{m} m_i * P \tag{4}$$

In this formula, P is a learned context model that estimates probabilities of coding terms, and a m is a mask to be applied to a latent variable representation of an input image in which coding terms are utilized.

While Mentzer may achieve impressive compression performance, we noticed that Mentzer appears to be more resource intensive with both longer training and inference times. However, we were intrigued by the importance map concept used in Mentzer and decided to augment Balle accordingly as described above.

5 Experiments/Results/Discussion

Multiscale Structural Similarity Index (MS-SSIM) is used to compare the quality of a compressed and subsequently reconstructed image to the original. MS-SSIM is widely used in current literature and is generally acknowledged to be an imperfect but sufficient substitute for how a human would view relative image quality [2] [1]. Other candidate metrics commonly used include Mean-Squared Error [8], and PNSR [4]. In addition to the image quality, the compression ratio, ratio of original to compressed image size, must be considered.

Method	Compression Ratio	MS-SSIM
JPEG 8/16/32	30.4, 21.5, 14.5	0.886, 0.945, 0.973
Balle Baseline	23.7	0.971
Balle Li Combination	26.5	0.968
Balle Encoder Variation	35.5	0.948
Balle Low/High Lambda	111.3, 9.1	0.885, 0.994
Balle Low/High Learn	21.1, 31.2	0.928, 0.723
Mentzer	Unavailable	0.944

Table 1: Average Compression Ratio and MSSSIM for 50 Image Test Set

Table 1 summarizes the three baselines and eight algorithm variations for which results were documented. Unless noted, algorithms were trained for 100,000 iterations with the ImageNet derived 20,000 image data set. All algorithms are evaluated using a 50 image data set containing a wide arrange of natural settings, man-made objects, people, and animals. These 50 test images range in size from 126 to 373 kB. The baselines are standard JPEG compression at three different quality settings.

The highest performing algorithm in terms of MS-SSIM was a version of the Balle algorithm with higher λ value which prioritized reducing the MSE error term in the loss function, however the compression ratio was much lower than other algorithms. Of the algorithms that achieved an average compression ratio of greater than 20, results for the three most promising are summarized in Figures 2 and 3. All three achieved better MS-SSIM values than JPEG 16 at a similar or better compression ratio. The Balle baseline with only minor modification achieved the best MS-SSIM of the three. **However**, our new combined Balle/Li approached achieved nearly as good MS-SSIM with significantly improved compression ratio, increase of 11.8%. Finally, 'Balle Encoder Variation' - for which we substantially adjusted Balle to have more layers / complex network and thus adjust the latent variable representation - achieved much higher compression, 35.5, than the others but with a greatly reduced MS-SSIM. However, it still outperformed JPEG 16 in terms of MS-SSIM despite having a 65% greater compression ratio. Figure 4 shows examples of the three promising compression schemes versus JPEG 16 for a specific image.

Figure 5, at left, shows the training loss history for five of the algorithm variations that were trained on AWS. Increasing and decreasing the learning rate by a factor of 10, shown in 'Learn+' and 'Learn-', caused poor performance. Especially in the case of the higher learning rate, where there occurred several large divergences. Seen at right in Figure 5, the compression rate is highly variable from image to image. Furthermore higher compression ratio generally correlated with lower MS-SSIM and smaller original image sizes.

6 Conclusion/Future Work

These results demonstrate several different approaches for neural network image compression that can produce superior results as compared to standard JPEG. Indeed, at the same compression rate one of the presented algorithms was able to provide 65.1 increased compression with the same quality as JPEG. Furthermore, a novel new combination of published approaches presented herein was able to perform at nearly the same quality as the best tested existing neural network algorithm (Balle Baseline) with significantly increased compression, demonstrating further increases in performance are possible.

While we made substantial progress on the following, we note that this was unable to be completed in time for submission. Modern smart phones and social networks may automatically identify certain contextual features (e.g., a user, the user's pets) in an image. We think it makes sense to use the bounding boxes of the Open Images dataset to train the importance map to allocate higher bitrate to these features known to be important to the user. Based on our research, this appears to be the first such contemplated use of this dataset

7 Figures

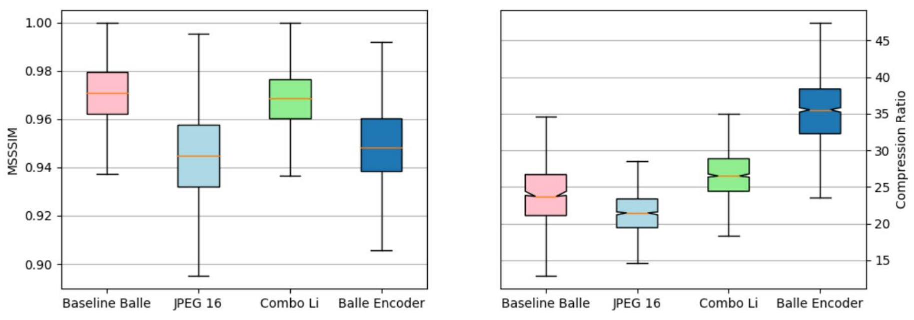

Figure 2: MS-SSIM and Compression Ratio Ranges for Most Promising Algorithms

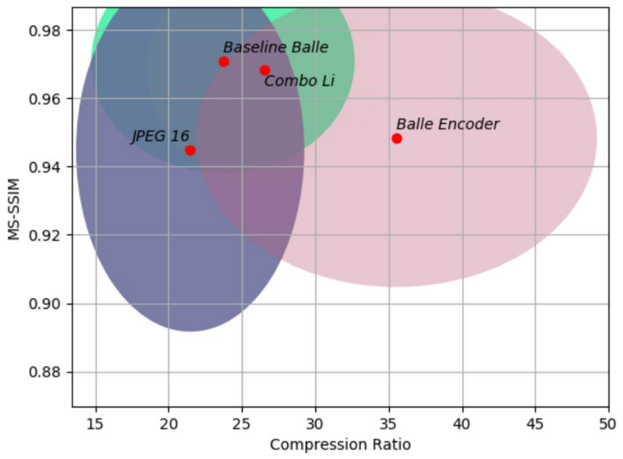

Figure 3: MS-SSIM and Compression Ratio Ranges for Most Promising Algorithms

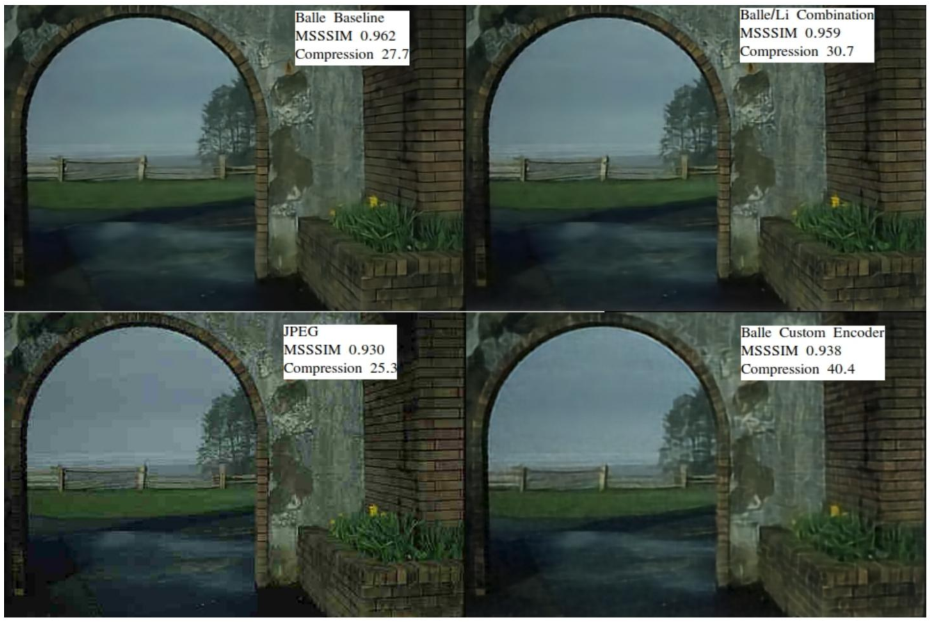

Figure 4: Example Reconstructed Images

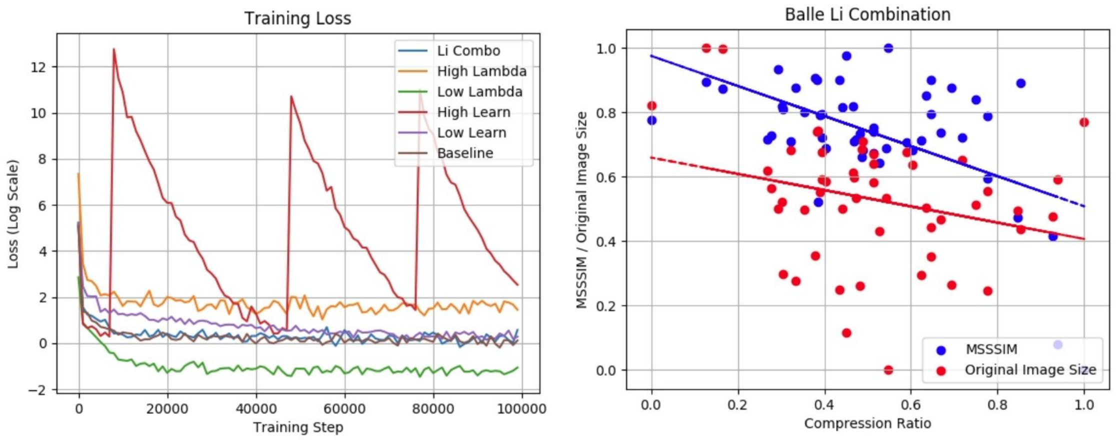

Figure 5: Training loss for AWS runs (left), example distribution of test results (right)

8 Contributions

Both team members shared responsibility for algorithm development and modifications, with Decoto focusing on Balle modifications and the Li algorithm, while Graham focused on implementation of the Mentzer algorithm, Open Images modifications, AWS training, and expansion of Balle algorithm features. Likewise, both team members shared equally in generation of reports and presentations.

References

[1] Feng Jiang, Wen Tao, Shaohui Liu, et all. *An End-to-End Compression Framework Based on Convolutional Neural Networks*. arXiv:1708.008.00838v1 [cs.CV], 2 August 2017.

[2] George Toderici, Damien Vincent, Nick Johnston, et all. *Full Resolution Image Compression with Recurrent Neural Networks*. arXiv:1608.05148v2 [cs.CV], 7 July 2017.

[3] Zihao Liu, Tao Liu, Wujie Wen, et all. *DeepN-JPEG: A Deep Neural Network Favorable JPEG Based Image Compression Framework*. arXiv:1803.05788v1 [cs.CV], 14 March 2018.

[4] Adnan Khashman, Kamil Dimililer. *Image Compression using Neural Networks and Haar Wavelet*. WSEAS Transactions on Signal Processing, ISSN: 1790-5052, Issues 5 Volume 4, May 2008.

[5] David Martin, Charles Fowlkes, Doron Tal, et al. *A Database of Human Segmented Natural Images and Application to Evaluation Segmentation Algorithms and Measuring Ecological Statistics*. ICCV Vancouver, July 2001.

[6] Jia Deng, Wei Dong, Richard Socher, et al. *ImageNet: A Large-Scale Hierarchical Image Database*. IEEE, June 2009.

[7] Johannes Ballé. *Efcient Nonlinear Transforms for Lossy Image Compression*. arXiv: 1802.00847v2 [eess.IV], 31 July 2018

[8] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte Luc Van Gool. *Generative Adversarial Networks GENERATIVE ADVERSARIAL NETWORKS FOR EXTREME LEARNED IMAGE COMPRESSION*. arXiv: 1804.02958v2 [cs.CV], 23 October 2018

[9] Johannes Balle, Valero Laparra, Eero Simoncelli. *End-to-End Optimized Image Compression*. arXiv: 1611.01704v3 [cs.CV], 3 March 2017

[10] Mu Li, Wangmeng Zuo, Shuhang Gu, et al. *Learning Convolutional Networks for Content-weighted Image Compression*. arXiv: 1703.10553v2 [cs.CV], 19 September 2017

[11] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, et al. *Conditional Probability Model for Deep Image Compression*. arXiv: 1801.04260v3 [cs.CV], 4 June 2018

[12] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, et al. *Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1*. arXiv:1602.02830v3 [cs.CV], 17 March 2016