12-12-2019

Extreme Image Compression with Deep Learning Autoencoder

Licheng Xiao

Santa Clara University

Department of Computer Science and Engineering

Date: December 12, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISIOR BY

Licheng Xiao

ENTITLED

Extreme Image Compression with Deep Learning Autoencoder

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

[illegible]

Thesis Advisor Dr. Nam Ling

Ying Liu

Thesis Reader Dr. Ying Liu

[illegible]

Chairman of Department Dr. Nam Ling

Extreme Image Compression with Deep Learning Autoencoder

By

Licheng Xiao

COEN 497

Master's Thesis Research

Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science and Engineering
in the School of Engineering at
Santa Clara University, 2019

Santa Clara, California, USA
Date: December 12, 2019

Dedicated to my family

Abstract

Image compression can save billions of dollars in the industry by reducing the bits needed to store and transfer an image without significantly losing visual quality. Traditional image compression methods use transform, quantization, predictive coding and entropy coding to tackle the problem, represented by international standards like JPEG (joint photographic experts group), JPEG 2000, BPG (better portable graphics), and HEIC (high efficiency image file format). Recently, there are deep learning based image compression approaches that achieved similar or better performance compared with traditional methods, represented by autoencoder, GAN (generative adversarial networks) and super-resolution based approaches.

In this paper, we built autoencoder based pipelines for extreme end-to-end image compression based on Ballé's approach in 2017 and 2018 and improved the cost function and network structure. We replaced MSE (mean square error) with RMSE (root mean square error) in the cost function and deepened the network by adding one more hidden layer before each strided convolutional layer. The source code is available in bit.ly/deepimagecompressiongithub.

Our 2018 approach outperformed Ballé's approach in 2018, which is the state-of-the-art open source implementation in image compression using deep learning in terms of PSNR (peak signal-to- noise ratio) and MS-SSIM (multi-scale structural similarity) with similar bpp (bits per pixel). It also outperformed all traditional image compression methods including JPEG, and HEIC in terms of reconstruction image quality. Regarding encoding and decoding time, our 2018 approach takes significant longer than traditional methods even with the support of GPU, this need to be measured and improved in the future.

Experimental results proved that deepening network in autoencoder can effectively increase model fitting without losing generalization when applied to image compression, if the network is designed appropriately.

In the future, this image compression method can be applied to video compression if encoding and decoding time can be reduced to an acceptable level. Automatic neural architecture search might also be applied to help find optimal network structure for autoencoder in image compression. Optimizer can also be replaced with trainable ones, like LSTM (long short-term memory) based optimizer. Last but not least, the cost function can also include encoding and decoding time, so that these two metrics can also be optimized during training.

Acknowledgements

I would like to thank Dr. Nam Ling, my advisor, for guiding me in research and paper writing, and Dr. Ying Liu for being the reader of my thesis.

I would also like to thank the authors whose papers were referenced in my thesis, their sharing of knowledge and software implementation provided me with a solid base to work on.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is Image Compression?

What is image? The ’image’ here refers to digital image, which is a two-dimensional plane with finite, discrete coordinates and intensity values. [1] In other words, digital image is a numeric representation of a two-dimensional image.

What is image compression? Image compression is a type of data compression applied to digital images. Data compression is the process of reducing the size of a data file. Therefore, image compression is the process of reducing the size of an image file or reducing the amount of data required to represent a digital image. [1]

1.2 Why is Image Compression Important?

Image compression can save the storage needed for storing digital images and videos, and save the bandwidth needed for transferring digital images and videos through the Internet. For example, to represent a two-hour movie of 720 × 480 pixels resolution in 24 bit RGB color domain using 30 fps (frames per second), the bandwidth needed is

$$30\frac{frames}{second} \times (720 \times 480)\frac{pixels}{frame} \times 24\frac{bit}{pixel} = 248{,}832{,}000\ bits/second = 248.832\ Mbps$$

Average global download bandwidth for fixed broadband is only 70.68 Mbps in October 2019, and the same bandwidth for mobile devices is only 38.23 Mbps, as shown in Figure 1.1.

The global Internet average download speed is significantly less than the bandwidth needed for real-time streaming of this standard definition (SD) video, not to mention high definition (HD) or even 4k videos. However, if a video was compressed by a compression ratio of 30, then the bandwidth needed would be 1/30, which is 8.2944 Mbps, and would be very easy to watch online in real time.

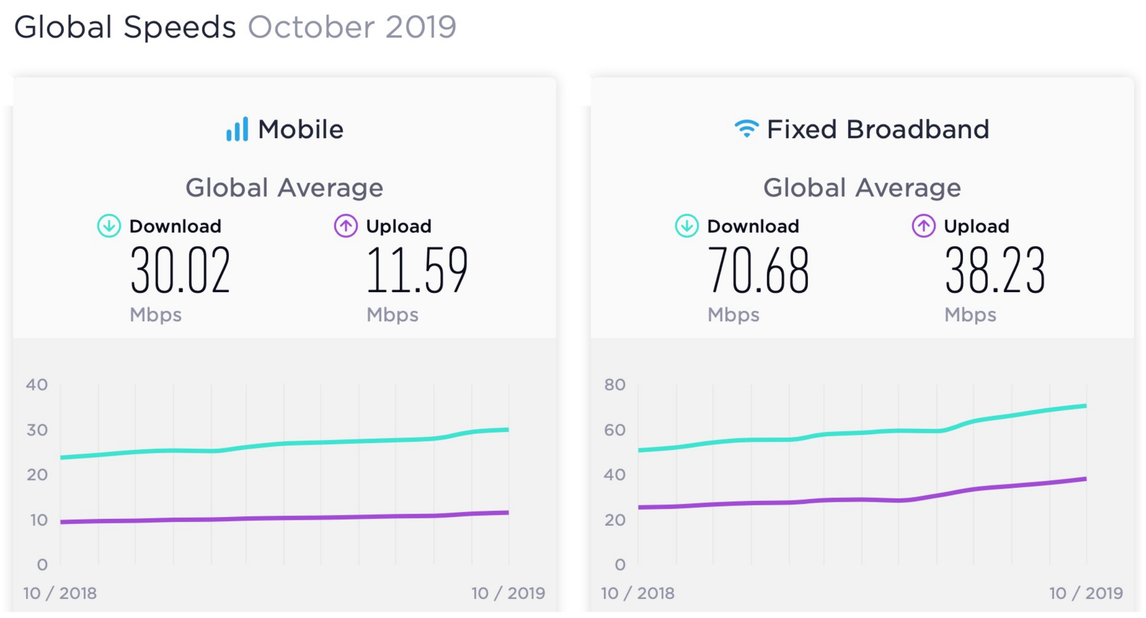

Figure 1.1: Average global Internet bandwidth from 2018/10 to 2019/10 [2]

Regarding storage, to store the SD movie mentioned above, the disk space needed is

$$248{,}832{,}000\frac{bits}{second} \div 8\frac{bits}{byte} \times (60^2)\frac{seconds}{hour} \times 2hour \cong 2.24 \times 10^{11} bytes = 224\ gigabytes$$

An uncompressed movie will consume almost all the storage available in a typical laptop, which often has only 256 GB storage. However, if the movie is compressed, a laptop can store many movies instead of just one.

1.3 Project Goal

The goal of the project is to improve the state-of-the-art image compression method and achieve the new state-of-the-art results. The previous state-of-the-art approach was proposed by Ballé et al in 2018 [3], and open sourced on github in 2019 [4].

1.4 Contributions

We successfully outperformed the previous state-of-the-art image compression method (Ballé et al 2018) in terms of PSNR (peak signal-to-noise ratio) and MS-SSIM (multi-scale structural similarity). With similar PSNR and MS-SSIM, our 2018 approach achieved around 5% rate saving.

We also proved that replacing single convolutional layer with double convolutional layers and smaller kernels before down sampling is an effective way in improving model performance of autoencoder models used in image compression.

Chapter 2

Literature

As we mentioned in our previous work [5], traditional image compression methods have a long history of development and are quite mature and complex now. From widely used JPEG to its successor JPEG2000 [6], and the latest state-of-the-art BPG [7], traditional lossy image compression methods have improved a lot in both reconstructive quality and compression ratio, it seemed hard to exceed BPG in a short time.

Different from traditional image compression methods, deep learning approach seems simpler. Deep learning approaches may use only deep neural networks to approximate any function it needs for image compression. By adjusting network structures, optimization goals, and optimization methods, the performance of such approximation can see improvement towards theoretical optimal status.

Regarding using deep learning to approximate lossy image compression, there have been several milestone research works conducted these years.

There are three major deep learning techniques used in image compression recently. One approach is generative adversarial networks (GAN). Another approach is super resolution using deep neural networks. The third approach, which is also the most important approach, is deep convolutional autoencoder.

GAN is good at generating images similar to original images, but it is not designed to minimize the difference in a controlled way. There are no explicit metrics to tell the difference between original images and reconstructed images, and the discriminator has to learn to distinguish arbitrary reconstructed images from all original images, which is much more complex than using autoencoder, which just need to compare one pair of images with explicit metrics. Experimental results have shown that GAN-based image compression often come with some visible difference in details, although it is impressive in compression ratio and image sharpness [8].

Super resolution using deep neural networks, represented by enhanced deep residual networks for single image super resolution (EDSR) [9], is the most limited approach because it is supervised learning, which needs both low-resolution image and corresponding high-resolution image as training data. Since low-resolution images are often achieved by traditional methods, such as bicubic down sampling, or using some deep neural networks that approximate bicubic down sampling, the accuracy of compressed data representation is limited by the traditional approach it used as reference, which makes it not as competitive as autoencoder in image quality, and not as competitive as GAN in compression ratio.

Deep convolutional autoencoder, represented by autoencoder applied to end-to-end image compression [10], outperformed the other two approaches, because its optimization goal can be set directly to minimize the difference between original image and reconstructed image, together with compressed file size, using metrics like mean square error (MSE), peak-signal-to-noise-ratio (PSNR), multi-scale structural similarity (MS-SSIM), and bits per pixel (bpp). This ensures that the deep neural network is always trained toward a correct direction. To the best of our knowledge, there are two deep convolutional autoencoder based approaches that already outperform BPG using deep learning. The first is by combining hierarchical entropy model with autoregressive priors [11]. The second is by context-adaptive entropy [12].

During our research, we focused on the state-of-the-art approach, which was proposed by Ballé's et al in 2017 [10] and then in 2018 [3]. These two approaches had their implementation open sourced in github in 2018 and 2019 [4]. These two approaches represented the state-of-the-art image compression approach using deep learning at the time they were published and could be easily reproduced leveraging the github implementations. Therefore, we selected these two approaches as our baseline 2017 and baseline 2018.

2.1 Baseline 2017 Overview

Ballé's approach in 2017 used an autoencoder structure to encode and decode images and use a trade-off of rates and distortion as the cost function to optimize the autoencoder. [10]

The high-level framework of the approach is shown in Figure 2.1. Note that the perceptual space is dedicated to gray scale images, and not applicable to color images. For color images, the distortion is calculated directly using Euclidean distance between original densities x and reconstructed densities $\hat{x}$ in the data space.

As is shown in Figure 2.1, $x \in R^N$ represents a vector of image intensities, and is mapped to a latent code space via a parametric analysis transform, $y = g_a(x; \varphi)$. y is quantized, yielding a discrete-valued vector $q \in Z^M$ which is then compressed. The rate of q, denoted by R, is lower-bounded by the entropy of the discrete probability distribution of the quantized vector, which is $H[P_q]$.

To reconstruct the compressed image, the discrete elements of q are reinterpreted as a continuous-valued vector $\hat{y}$, which is transformed back to the data space using a parametric synthesis transform $\hat{x} = g_s(\hat{y}; \theta)$.For gray-scale images, the distortion is assessed by transforming

$\hat{x}$ to a perceptual space using a (fixed) transform, $\hat{z} = g_p(\hat{x})$, and evaluating a metric $d(z, \hat{z})$. For color images, the distortion is assessed by $d(x, \hat{x})$ directly.

Ballé' et al optimize the parameter vectors ϕ and θ for a weighted sum of the rate and distortion measures, R + λD, over a set of images.

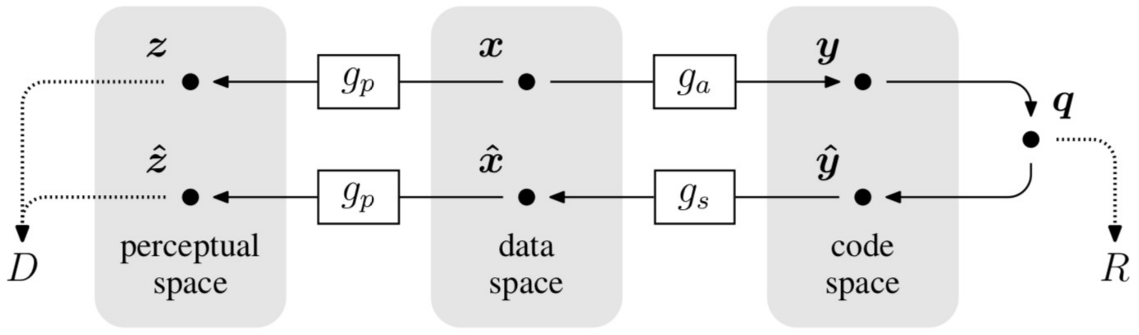

Figure 2.1: High-level framework of Ballé's approach in 2017. [10]

The structure of the autoencoder is shown in Figure 2.2. The structure is quite similar to classic deep autoencoder based on CNN. Shallow depth and simple structure made it fast to compress and decompress images.

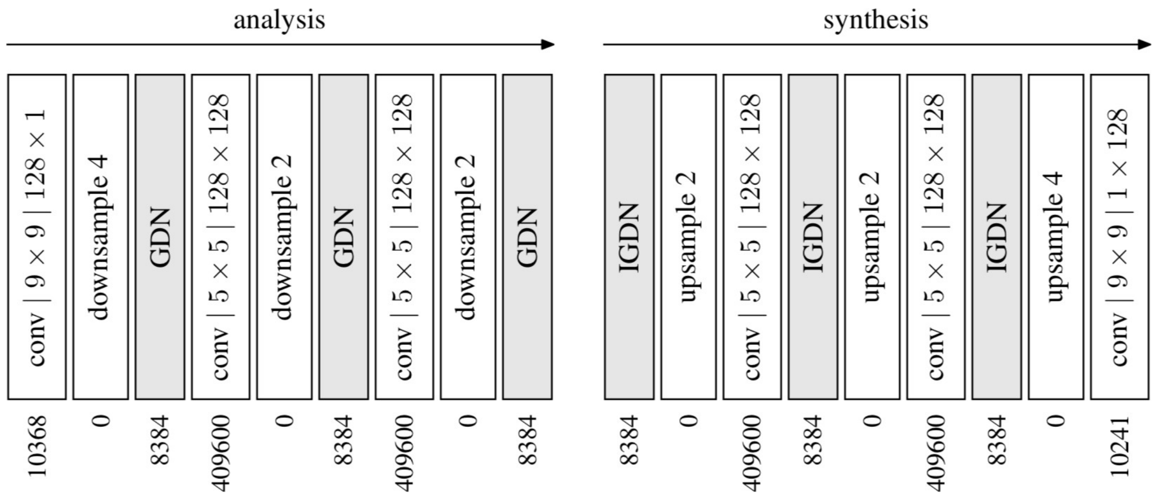

Figure 2.2: Network structure of the autoencoder used in Ballé's approach in 2017. [10]

The most special component in Ballé's approach in 2017 different from ordinary deep autoencoder is GDN (general divisive normalization). GDN (general divisive normalization) is a parametric nonlinear transformation that is well-suited for Gaussianizing data from natural images. It is essentially a multivariate generalization of a particular sigmoid-type function, as described in Equation (1). [13]

$$y = g(x;\theta) \qquad s.t. \qquad y_i = \frac{z_i}{\left(\beta + \sum_j \gamma_{ij} |z_j|^{\alpha_{ij}}\right)^{\varepsilon_i}} \qquad and \qquad z = Hx \tag{1}$$

where H is a whitening matrix used to de-correlate two-dimensional densities, which is similar to ZCA (zero-phase component analysis) [14]. β, γ, α and ε are all trainable parameters that help to generalize the transformation.

When implemented in baseline 2017, α and ε were fixed on purpose, and the transformation can be described as Equation (2). This implementation never sums across spatial dimensions. It is similar to local response normalization, but much more flexible, as β and γ are trainable parameters. [4]

$$y = g(x;\theta) \qquad s.t. \qquad y_i = \frac{x_i}{\sqrt{\beta + \sum_j \gamma_{ij} {x_j}^2}} \tag{2}$$

2.2 Baseline 2018 Overview

Baseline 2018 was built on top of baseline 2017, and the major difference is that it incorporates a hyperprior to capture spatial dependencies in the latent representation. This hyperprior relates to side information and was trained jointly with the underlying autoencoder [3]. This innovation yielded the state-of-the-art rate-distortion performance so far in all published ANNs (artificial neural networks) with open source implementation.

The structure of baseline 2018 is shown in Figure 2.3. The left side shows an image autoencoder architecture, similar to baseline 2017. The right side corresponds to the autoencoder implementing the hyperprior. The factorized-prior model uses the identical architecture for the analysis and synthesis transforms g_a and g_s. Q represents quantization. AE and AD represent arithmetic encoder and arithmetic decoder, respectively. Convolution parameters are denoted as: number of filters × kernel support height × kernel support width / down or up sampling stride, where ↑ indicates up sampling and ↓ down sampling. N and M were chosen dependent on λ [3]. For baseline 2018, I chose N = M = 192 when λ = 0.01.

The encoder subjects the input image x to g_a, yielding the responses y with spatially varying standard deviations. The responses are fed into h_a, summarizing the distribution of standard deviations in z. z is then quantized, compressed, and transmitted as side information. The encoder then uses the quantized vector $\hat{z}$ to estimate $\hat{\sigma}$, the spatial distribution of standard deviations, and uses it to compress and transmit the quantized image representation $\hat{y}$. The decoder first recovers $\hat{z}$ from the compressed signal. It then uses h_s to obtain $\hat{\sigma}$, which provides it with the correct probability estimates to successfully recover $\hat{y}$ as well. It then feeds $\hat{y}$ into g_s to obtain the reconstructed image. [3]

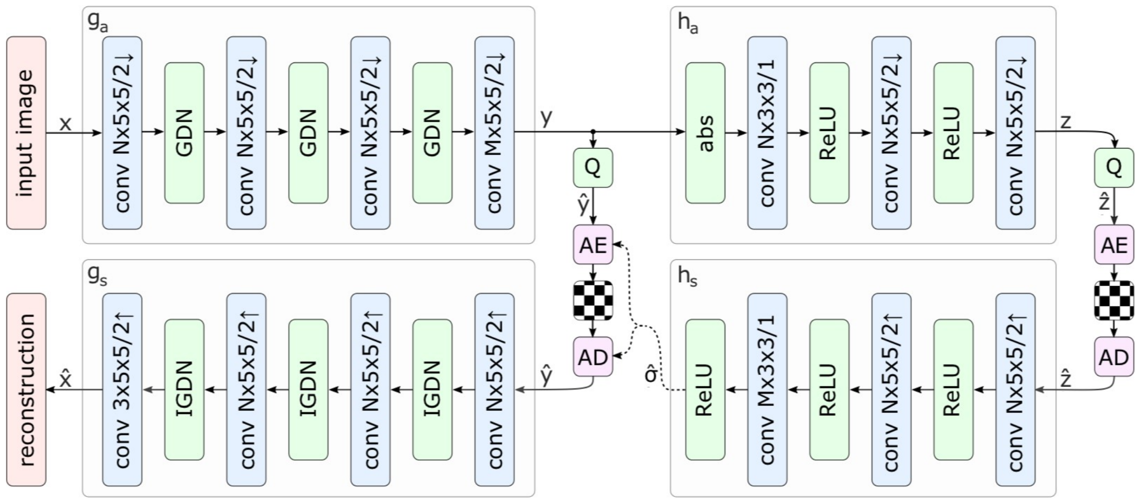

Figure 2.3: Network architecture of baseline 2018. [3]

Chapter 3

Proposed Methods

We experimented two major groups of improvements on Ballé's approach. One is based on Ballé's approach in 2017, and the other is based on Ballé's approach in 2018.

3.1 Improvement on the 2017 Baseline

As mentioned in the previous work [5], we propose an improved deep convolutional autoencoder for end-to-end lossy image compression, which is optimized for RMSE (root mean square error) and bpp (bits per pixel), and also has deeper and doubled network structure. Our 2017 approach is different from previous work that set optimization goals only on MSE (mean square error) and bpp [10], or MS-SSIM and bpp [3], or a choice between optimal MSE and optimal MS-SSIM (multi-scale structural similarity) [12]. Experimental results showed that our proposed approach outperformed previous state-of-the-art artificial neural network (ANN)-based image compression approaches represented by Ballé [10], as well as BPG (RGB 4:4:4).

3.1.1 Cost Function

One contribution of our work is the improved optimization goal. Since optimization goal has substantial influence on the performance of a trained ANN, I'm especially interested in modifying the optimization goal, and look for better performance. As proved by previous experiments by Ballé [3] if the autoencoder was optimized for MSE, the result was better than JPEG2000, and just slightly weaker than BPG. If the autoencoder was optimized for MS-SSIM, the result was similar in overall fidelity, but varied for images with different local contrast. Although experimental results exceeded their previous ANN approaches, none of them exceeded BPG. Even in the latest two approaches that outperformed BPG, none of these results were achieved by improving the optimization goal. Here we tried to improve the optimization goal.

Optimization goal, i.e. objective function or loss function of a variational autoencoder [10] can be expressed as Equation (3), with the total loss (denoted by L), of encoder (denoted by g_a), decoder (denoted by g_s), and the discrete probability distribution of the quantized vector (denoted by P_q).

When encoding the input image (denoted by x), the input image x is first transformed by encoder g_a into a continuous-valued vector (denoted by y in Figure 3.1, Figure 3.2, and Figure 3.3), and then quantized into a discrete-valued vector (denoted by q) through a uniform scalar quantizer, which rounds each element to the nearest integer (denoted by Q), and finally use an entropy encoder to encode the quantized vector q to generate the compressed file.

When decoding the compressed file, we first use an entropy decoder to decode the compressed file into quantized vector q, then reinterpret the quantized vector q from discrete-valued vector to continuous-valued vector (denoted by y_tilde in Figure 3.2 and Figure 3.3) using interpreter function (denoted by I), then use decoder g_s to synthesis reconstructed image (denoted by $\hat{x}$).

The loss function equals to the entropy of the discrete probability distribution of the quantized vector, plus the distance (denoted by d) between input image x and reconstructed image $\hat{x}$ multiplied by Lagrange multiplier lambda (denoted by λ).

$$\begin{aligned} &L\,[g_a,\, g_s,\, P_q] = -E\,[\log_2(P_q)] + \lambda E\,[d\,(x,\,\hat{x})] \\ &where\ q = Q(g_a(x)) \\ &and\ \hat{x} = g_s(I(q)) \end{aligned} \tag{3}$$

In our 2017 approach, we leave the entropy part unchanged, and modify the distance part. Originally in Ballé's design [10], the distance function is just the MSE of perceptual space, which can be expressed as Equation (4), where x denotes input image and $\hat{x}$ denotes reconstructed image. The problem of using MSE as the measurement of distance is that if the model makes a single very bad prediction, the squaring will make the error even worse and it may skew the metric towards overestimating the model's badness. That is a particularly problematic behavior if there are lots of noise in the training data. On the other hand, if all the errors are small, or rather, smaller than 1, then the opposite effect is felt: the model's badness may be underestimated.

$$d_0(x,\hat{x}) = MSE = \|x,\hat{x}\|_2^2 \tag{4}$$

In our 2017 approach, we use RMSE instead of MSE as the measurement of distortion as in Equation (5). The major advantage of RMSE is that it is at the same scale as the target. In case of noisy data, RMSE is much smaller than MSE, which can prevent overestimating the model's badness, and in case of small error less than 1, RMSE is much larger than MSE, which can prevent underestimating the model's badness.

$$d_1(x,\hat{x}) = RMSE = \sqrt{\|x,\hat{x}\|_2^2} \tag{5}$$

3.1.2 Network Structure

The original analysis transform network and synthesis transform network designed by Ballé [10] each contains only 3 convolutional layers, with kernel size of 9× 9, 5× 5 and 5× 5, as shown in Figure 3.1, where ↓ denotes down sampling, ↑ denotes up sampling, GDN stands for generalized divisive normalization, and IGDN stands for inverse generalized divisive normalization. GDN and IGDN are inspired by models of neurons in biological visual systems and are effective in Gaussianizing image densities [13]. Besides, y denotes the transformed continuous-valued vector; it is then quantized by a uniform scalar quantizer (i.e. each element is rounded to the nearest integer) to generate the quantized vector q, and y_tilde denotes a continuous-valued vector reinterpreted from discrete elements of quantized vector q, as mentioned in Equation (3).

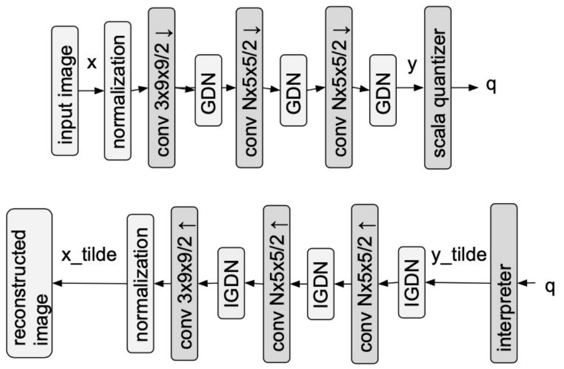

Figure 3.1: Ballé’s network structure for analysis transform (encoder, on the top) and synthesis transform (decoder, at the bottom). [10]

Inspired by progressive growing generative adversarial network (PGGAN) [15], we modify the network structure of autoencoder-like transformer, to make it deeper. As is shown in Figure 3.2, we reduce the kernel size to 9 × 9, and add three more layers in the encoder, i.e. analysis transform. The benefit of doing this is that the encoder is capable to capture larger spatial features as the network become deeper. However, there is also minor disadvantage for doing so, which is the information loss during down sampling, which may cause it harder to preserve enough information before entropy coding. The decoder part, i.e. synthesis transform, has a similar structure as the encoder, but with opposite directions.

However, single layer of convolution does not provide enough trainable parameters and structures to learn complex feature representations before downscaling. That is why PGGAN uses two fully connected convolutional layers before each down sampling step. Therefore, we doubled the convolutional layers before each GDN and IGDN in our structure, as in Figure 3.3, which allows the network to learn more features before losing information during down sampling.

As PGGAN outperforms previous GAN approaches in generative tasks, we expect such deeper

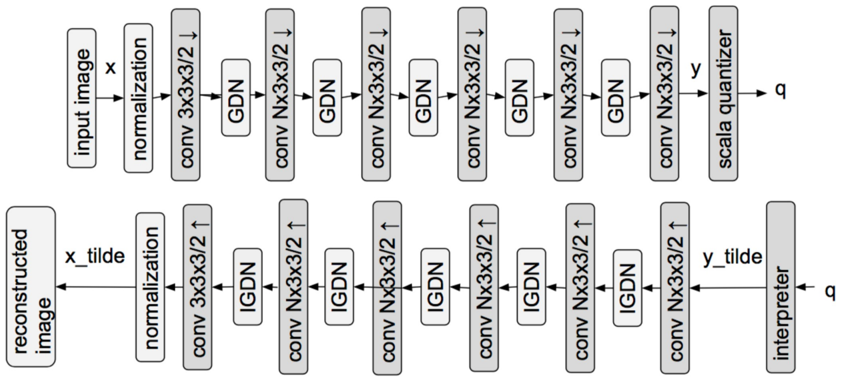

Figure 3.2: Our deeper network structure for analysis transform (encoder, on the top) and synthesis transform (decoder, at the bottom).

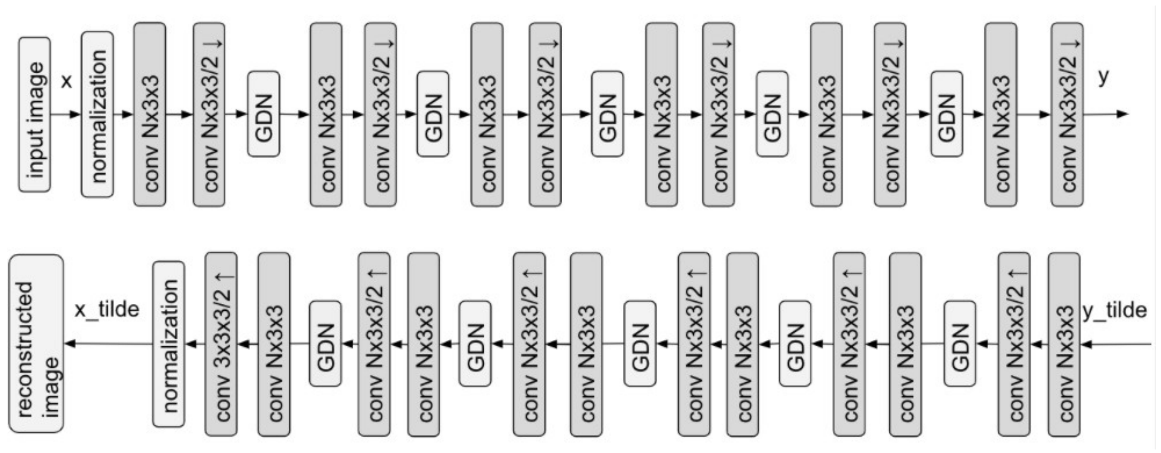

Figure 3.3: Our deeper and doubled network structure for analysis transform (encoder, on the top) and synthesis transform (decoder, at the bottom).

network structure would also boost the performance of autoencoder, which was supported by experimental results.

Although not directly compared in this paper, the experimental results of the deeper network design in Figure 3.2 and Figure 3.3 were better than that of the previous shallower network structure in Figure 3.1, which can be concluded from the improvements of our 2017 approach over BPG (RGB 4:4:4), which was not achieved by Ballé's approach [10].

3.2 Improvement on the 2018 Baseline

During improving the 2017 baseline, we did too many experiments, and did not record all the intermediate results, which made it difficult to isolate the influence of each variable. Therefore, when improving the 2018 baseline, we reduced the number of variables that we modified simultaneously, so that we can better track the differences caused by changing single variable. we used very similar network structure as the 2018 baseline, only replaced the kernel size from 5× 5 to 3× 3 and added one additional convolutional layer before each down sampling or up sampling. Experimental results shown that our modifications improved the compression ratio by around 5% with similar reconstruction quality.

3.2.1 Network Structure

Similar to our modifications on the 2017 baseline, we added one more convolutional layer before each down sampling or up sampling. The difference is that we used the same number of down sampling and up sampling as that of the 2018 baseline, to better isolate variables in our experiments. Since the encoder and decoder were similar in structure, we use encoder as example to illustrate our improvement as in Figure 3.4. In the encoder of Ballé's approach in 2018, there are 4 convolutional layers with kernel size of 5 × 5 and 4 down sampling.

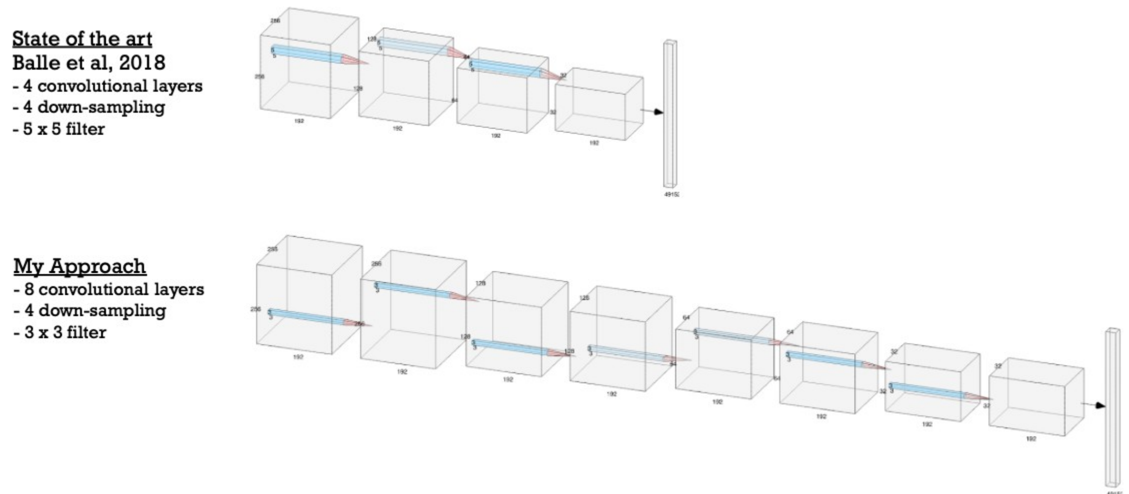

Figure 3.4: Comparison of encoder structure of Ballé's approach in 2018 and our 2018 approach.

This modification is the best among multiple temptations for structural improvement. Experimental results have shown that this modification works well and can provide around 5% improvement on bpp with similar MSE when the models were trained to 1 million iterations.

The reason why our modifications works better is possibly that more convolutional layers before information loss caused by down sampling can provide more parameters to learn image features, so that the model can better capture details in image compression. This is similar to our improvement on the 2017 baseline, but with more strict control on independent variables.

The size of kernels was reduced from 5×5 to 3×3 to maintain a relatively similar scale for overall convolution, since the model has 2 convolutional layers instead of 1 before each down sampling or up sampling in our 2018 approach. This should not have much influence on the result but can be isolated in future experiments to be better proved.

Chapter 4

Experimental Results

Two groups of experiments were conducted; one was based on Ballé's approach in 2017 and the other was based on Ballé's approach in 2018.

4.1 The 2017 baseline and Our Improvement

Our modifications and experiments are based on a Python implementation [4] of Ballé's work in 2017 [10].

We trained the autoencoder with 139 high resolution sRGB images of natural scenes from CLIC (challenge on learned image compression) [16] professional training dataset. Each image was cropped to 1024 × 1024. For each iteration the model was trained with a batch of eight randomly cropped 256 × 256 patches truncated to RGB format. For quick comparison between different model configurations, the early stage models were trained for only 10K iterations, and there had been hundreds of such quick and small experiments. Then several outstanding models were selected, and continued to be trained to 50K iterations, and came to the second round of selection. The third round was 250K iterations, and the final stage models were trained to 1M iterations.

Our 2017 approach was evaluated using Kodak True Color Image Suite [17], for the average of PSNR, MS-SSIM, and bpp on all 24 images, in comparison with BPG (RGB 4:4:4). The results showed that our proposed approach with improved optimization goal as described in Equation (5) and deeper network structures as described in Figure 3.3 has achieved better performance than BPG (RGB 4:4:4) in terms of subjective fidelity in sharpness and details.

4.1.1 Multi-point Comparison between BPG and Our 2017 Approach

We trained several models with different λ's, and compared them with BPG (RGB 4:4:4) on PSNR and MS-SSIM at comparable bpp, and approximated two curves on a two-dimensional panel using linear regression. Our 2017 approach is superior in both metrics for all these configurations of λ.

From Figure 4.1 one can see that the PSNR of our 2017 approach (red line) is consistently higher than that of BPG (RGB 4:4:4) by around 1%, when bpp is raised from 0.1 to 0.4. Although PSNR under 30 is still not considered good quality, it is acceptable for the compression ratio between 60 and 240.

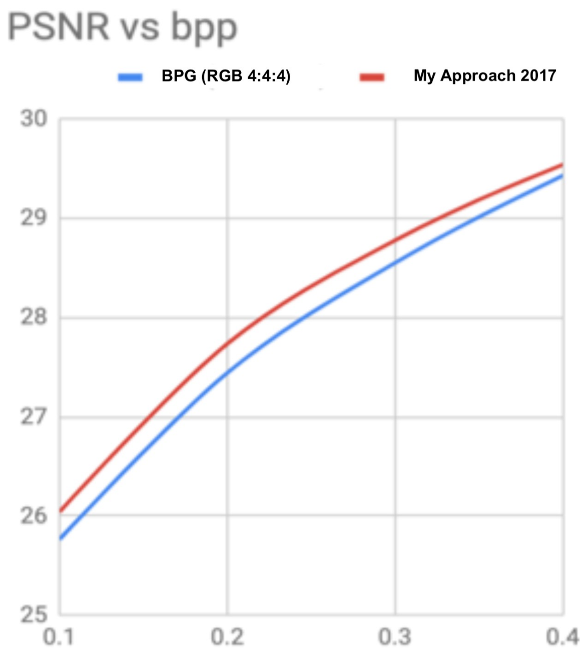

Figure 4.1: PSNR (dB) vs bpp on Kodak dataset. The vertical axis represents average PSNR over 24 Kodak images, the horizontal axis represents average bpp (bits per pixel) over 24 Kodak images.

The most impressive improvement is on MS-SSIM, which one can see from Figure 4.2 that the MS-SSIM of our 2017 approach (red line) is consistently higher than that of BPG (RGB 4:4:4) by around 23% at the same bpp. This means that our 2017 approach can preserve significantly more structural information than BPG (RGB 4:4:4) with the same compression ratio.

4.1.2 Single-point Comparison between the 2017 Baseline and Our 2017 Approach

When comparing the baseline (Ballé's approach in 2017) and our 2017 approach, we found that our 2017 approach can achieve higher reconstruction image quality measured by lower MSE (directly related to PSNR) with similar bpp (bits per pixel).

As is shown in Figure 4.3, our 2017 approach has higher bpp at the beginning of the training process comparing with Ballé's approach in 2017. However, the bpp of our 2017 approach quickly

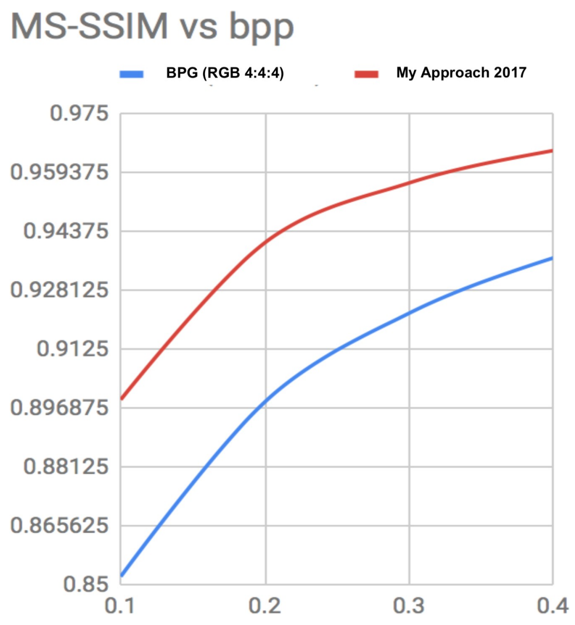

Figure 4.2: MS-SSIM vs bpp on Kodak dataset. The vertical axis represents average MS-SSIM over 24 Kodak images, and the horizontal axis represents average bpp (bits per pixel) over 24 Kodak images.

decreased as time went on and reached the same level as that of Ballé's approach in 2017 at iteration 550K.

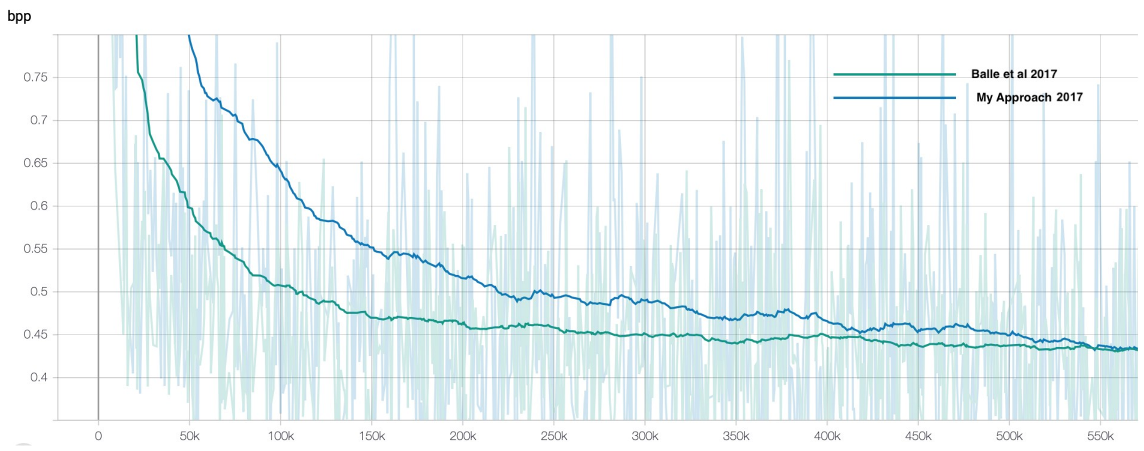

Figure 4.3: Comparison of estimated bpp over training iterations between Ballé's approach in 2017 and our 2017 approach before 550K iterations.

As is shown in Figure 4.4, our 2017 approach has higher MSE (mean square error) at the beginning of the training process comparing with Ballé's approach in 2017. However, the MSE of our 2017 approach quickly decreased as time went on and reached the same level as that of Ballé's approach in 2017 at around iteration 150K and reached a significantly lower level than that of Ballé's approach in 2017 at iteration 550K, when the bpp of both approaches were at the same level.

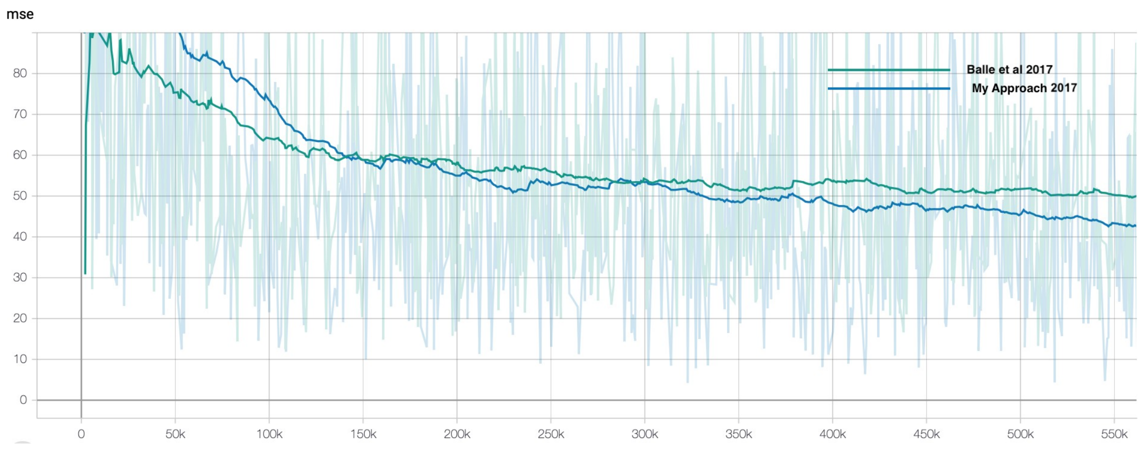

Figure 4.4: Comparison of estimated MSE (mean square error) over training iterations between Ballé's approach in 2017 and our 2017 approach before 550K iterations.

Both bpp and MSE used in the above comparisons were smoothed values, which can better demonstrate the overall trend and average performance of the two metrics.

One interesting observation was that, after 550K iterations, the performance improvement of our 2017 approach seemed to slow down and reached a ceiling comparing with that of Ballé's approach in 2017. As can been seen in Figure 4.5, the bpp of our 2017 approach didn't decrease a lot after 550K iterations, and the same phenomenon happened to MSE which can be seen in Figure 4.6.

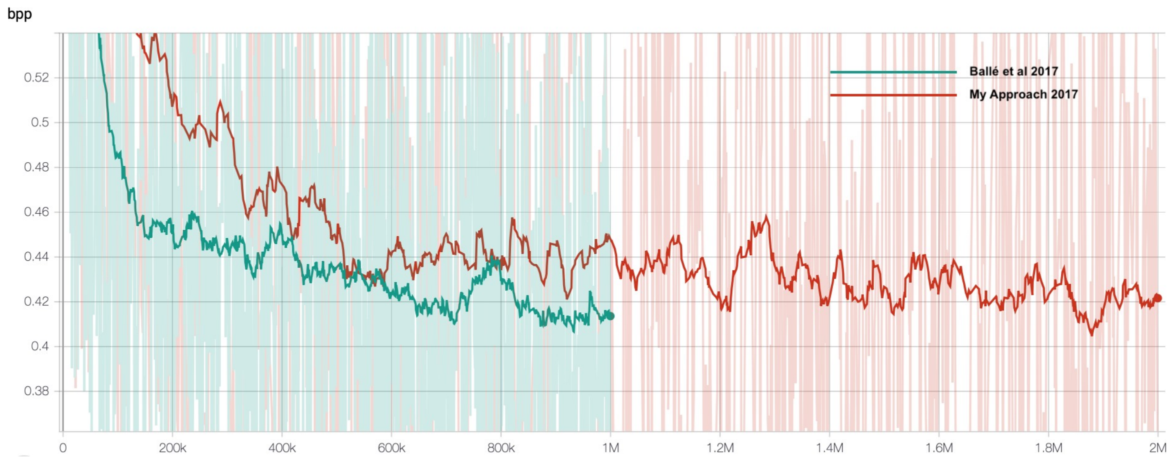

Figure 4.5: Comparison of estimated bpp over training iterations between Ballé's approach in 2017 and our 2017 approach before 2M iterations.

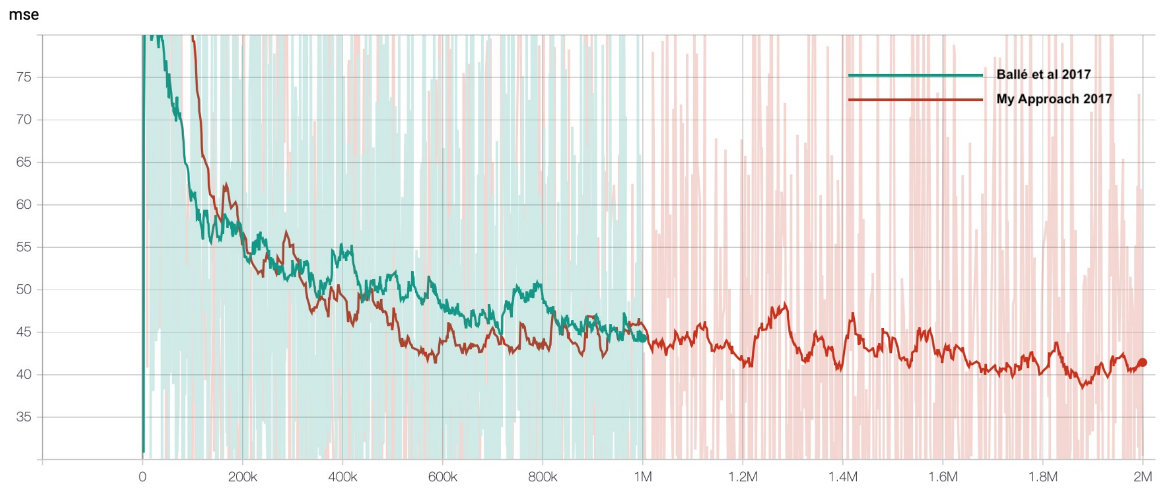

Figure 4.6: Comparison of estimated MSE (mean square error) over training iterations between Ballé's approach in 2017 and our 2017 approach before 2M iterations.

One possible reason for this bottleneck phenomenon of our 2017 approach is too many down samplings comparing with the 2017 baseline. The 2017 baseline has only three down samplings in the encoder, but our 2017 approach has 6 down samplings. These obsessive down samplings caused our 2017 approach to lose too much information comparing with the 2017 baseline so that it reached a performance ceiling after impressive performance before 550K iterations.

4.1.3 Image Examples

Figure 4.7 shows one of the original lossless pictures from Kodak True Color Image Suite, and one can see clear details of the trees on the top right corner and the wall tiles on the bottom right corner, especially the transition of shadows between trees (marked with a blue box), and the sharp edges of each wall tile (marked with a blue box), which was compared to the compressed pictures later.

Figure 4.8 shows the picture compressed by BGP (RGB 4:4:4), and one can see significant blurred effect on the trees and wall tiles when comparing with the original picture. The shadows of the trees become flat blocks (marked with a red box), and the edges of the wall tiles almost disappear (marked with a red box). However, the bpp of this compressed picture is 0.835, relatively higher than that of our 2017 approach which is shown next.

Figure 4.9 shows the picture compressed by our 2017 approach, where the details of the trees and wall tiles (marked with green boxes) are well preserved when comparing with the original picture, and is much sharper and clearer than that of the picture compressed by BPG (RGB 4:4:4), even though the bpp is at only 0.561, significantly lower than that of BPG (RGB 4:4:4).

The major reason for such significant improvement in sharpness and detail fidelity is that our 2017 approach provides a non-linear transformer implemented as deep neural network which is

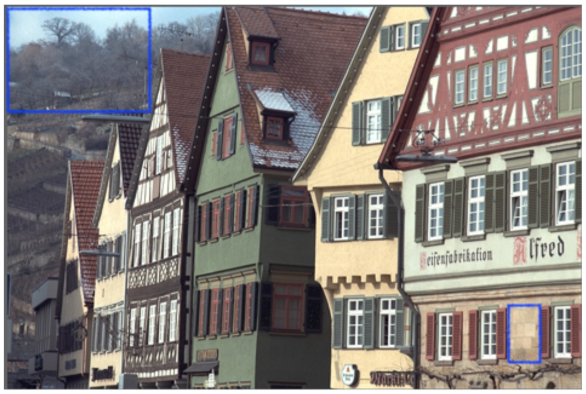

Figure 4.7: Original picture. Blue box arears are used to compare compression ratio of BPG (RGB 4:4:4) and that of our 2017 approach.

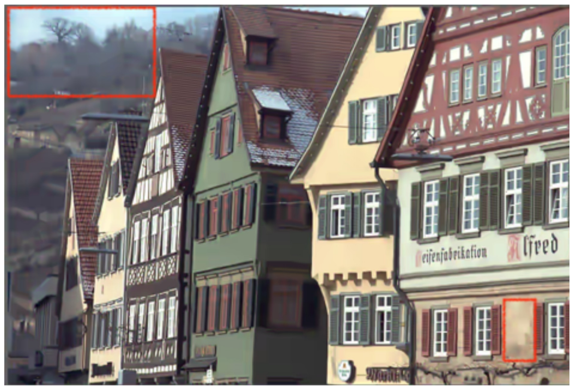

Figure 4.8: Picture compressed with BPG (RGB 4:4:4), QP = 42, compression level = 9, bpp = 0.835. Red box areas look blurred.

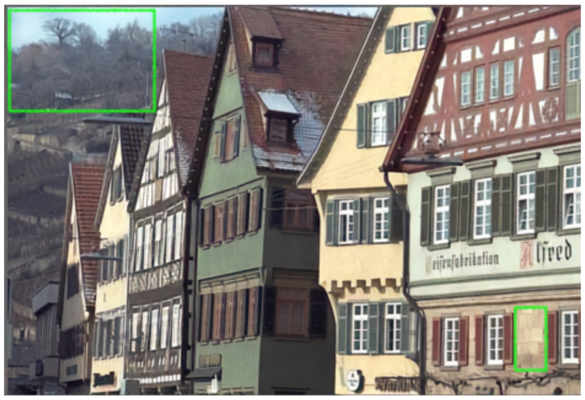

Figure 4.9: Picture compressed with our 2017 approach, bpp = 0.561. Green box areas are still very sharp.

better than the linear integer transformer used in BPG in terms of transforming from intensity domain to frequency domain.

In plain areas, such as pure color walls, there are not much difference regarding color or luminance between the original picture, the picture compressed by BPG (RGB 4:4:4) and that by our 2017 approach. Both BPG (RGB 4:4:4) and our 2017 approach performed very well on these plain areas.

4.2 The 2018 Baseline and Our Improvement

For the 2018 baseline, we only trained the model with lambda = 0.01 with 1M iterations. Each experiment took around 1 week on single GPU (Nvidia GeForce GTX 1070).

All models were trained using CLIC professional-train dataset [16] and evaluated using CLIC professional-valid dataset. The CLIC dataset contains around 2000 pictures in total, and the professional sub-dataset contains around 600 pictures. The pictures in the professional sub-dataset are of different categories, including sceneries in the wild, buildings in the city, portraits, and indoor furnitures. The pictures were taken using professional cameras, with resolutions from standard definition to 2K high definition, in sRGB color domain.

The 2018 baseline used RGB domain as the only supported color domain, and we did not change that part during all our experiments. Therefore, during preprocessing, we converted all the training images from sRGB domain to RGB domain.

Each training image was randomly sliced to patches of size 256 × 256. Ideally, this step belongs to preprocessing and should be executed independent of the training process. However, the baseline script directly included this step in the training process, which might have some influence on duplicating experimental results. Considering the training iterations was more than 1 million, which was much larger than the number of training images, the differences caused by randomly slicing should be minimal and can be ignored. Therefore, we did not change this part in our 2018 approach during these experiments.

Among all models we trained, the best one is the one described in Section 3.2.1, which added one additional convolutional layer before each down sampling or up sampling with kernel size of 3 × 3 instead of 5 × 5.

4.2.1 Single-point comparison between the 2018 Baseline, Our 2018 approach, JPEG2000 and HEIC

We trained both the 2018 baseline and our 2018 approach to 1 million iterations and observed that our 2018 approach can achieve significantly lower bpp (bits per pixel) with similar MSE (mean square error), and this trend is consistent and stable throughout the training process.

As is shown in Figure 4.10, at the beginning, the MSE (mean square error) of our 2018 approach is higher than that of Ballé's approach in 2018. However, as training went on, the MSE of our 2018 approach steadily decreased to the same level as that of Ballé's approach in 2018 and reached a stable status from 600k to 1M iterations.

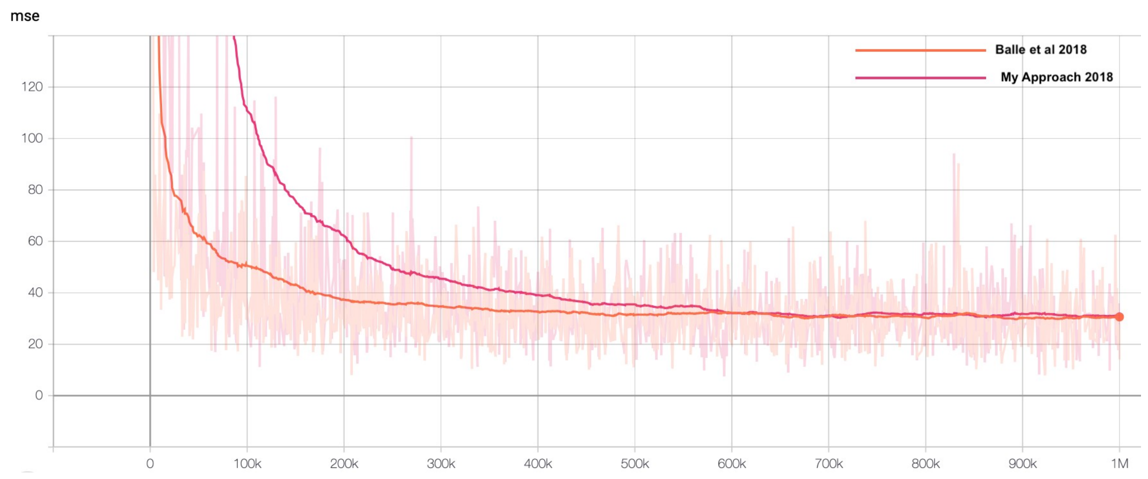

Figure 4.10: Comparison of MSE (mean square error) over training iterations between Ballé's approach in 2018 and our 2018 approach.

As is shown in Figure 4.11, at the beginning, the bpp (bits per pixel) of our 2018 approach was lower than that of Ballé's approach in 2018, and as training went on, this advantage maintained well till the end of 1 million iterations. At the end of training, our 2018 approach achieved around 5% reduction in bpp (bits per pixel) than that of Ballé's approach in 2018.

Note that the first 50K iterations were a bit different than later iterations, as is shown in Figure 4.12. Our 2018 approach started with a lower bpp, and Ballé's approach in 2018 started with a higher bpp. From iteration 15k to 30k, the bpp of both approaches were very close to each other. After iteration 30k, the bpp of our 2018 approach started to decrease faster than that of Ballé's approach in 2018. The reason for this difference in the starting bpp still needs further experiments to reveal.

Averaged on all 41 pictures in CLIC professional validation dataset, our 2018 approach achieved better performance than all previous standards, as is shown in Table 4.1. Compared with baseline 2018, our 2018 approach achieved around 5.4% reduction in bpp, 2.2% increase in PSNR, and 3.6% increase in MS-SSIM.

Note that the HEIC results were incomplete because some compressed images were different in resolution comparing with the original images so that it's not feasible to calculate PSNR and MS-SSIM for these images.

Besides, the encoding time and decoding time for JPEG2000 and HEIC were estimated

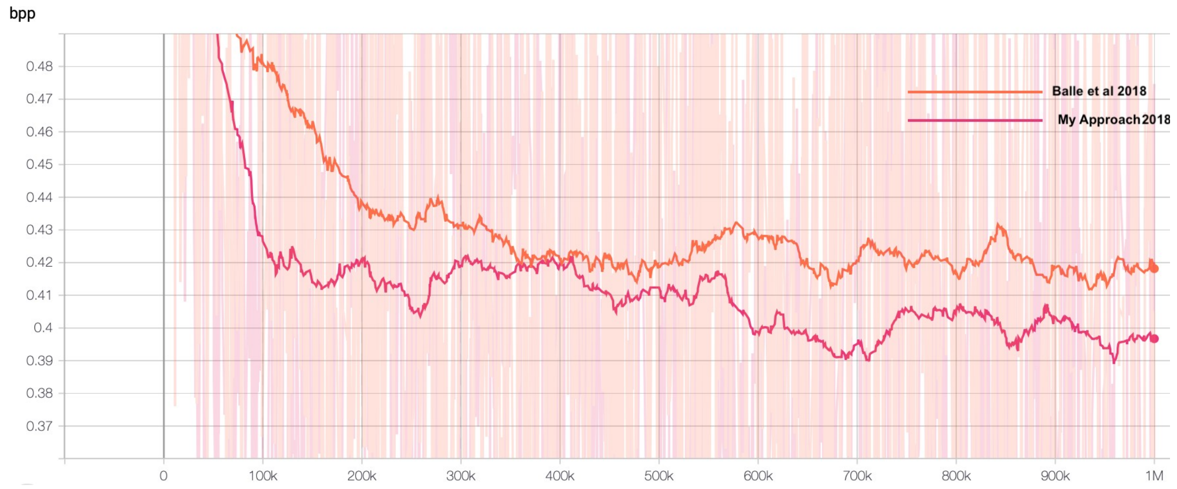

Figure 4.11: Comparison of bpp (bits per pixel) over training iterations between Ballé's approach in 2018 and our 2018 approach.

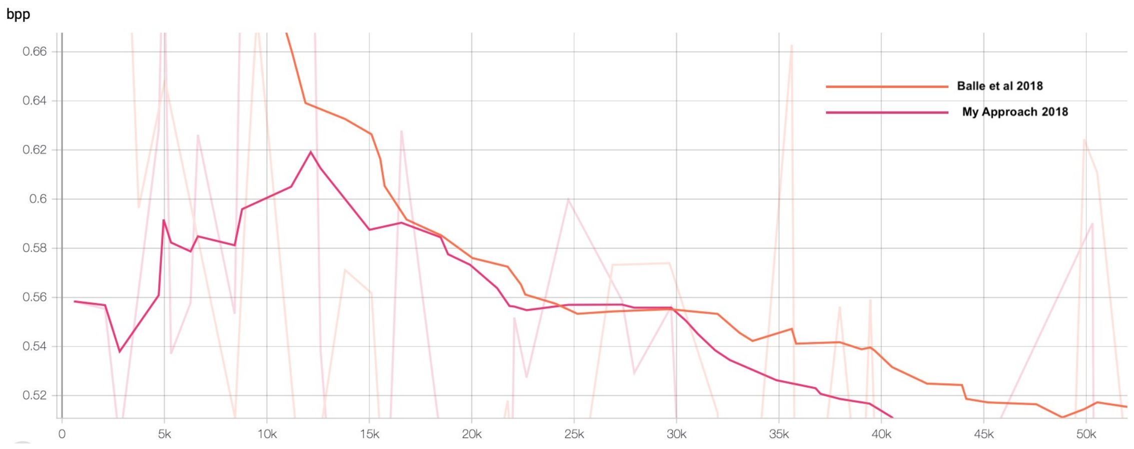

Figure 4.12: Comparison of bpp (bits per pixel) in the first 50K training iterations between Ballé's approach in 2018 and our 2018 approach.

manually by encoding and decoding all 41 images in CLIC professional validation dataset on a MacBook Pro laptop. The actual time should be slightly shorter since the I/O time was not subtracted from SSD (solid state drive).

For the baseline 2018, the official script did not include timing options, but we added it by ourselves and also to our 2018 approach. We pre-loaded the model before encoding and decoding images and excluded I/O time when calculating encoding and decoding time for the baseline 2018 and our 2018 approach. Since images in CLIC professional validation dataset were usually in high definition (2K), the encoding and decoding time were much higher than traditional codecs. However, for smaller images like those in Kodak dataset with a resolution of 512 × 768, the encoding and decoding time were similar to traditional methods.

Note that the encoding time for our 2018 approach was significantly prolonged due to exceeding GPU memory limit for some large images. This time penalty should disappear given larger GPU memory or smaller input image resolutions.

Table 4.1: Comparison between JPEG2000, HEIC, Ballé's approach in 2018, our 2018 approach, over CLIC professional validation dataset.

Standard	Average bpp	Average PSNR	Average MS-SSIM	Encoding Time (seconds)	Decoding Time (seconds)	Hardware
JPEG2000	0.5154	14.01	0.8443	0.2	0.18	Intel Core i7 2.8Hz 2 core
HEIC	0.1548	12.32	0.8313	0.23	0.20	Intel Core i7 2.8Hz 2 core
Ballé et al 2018	0.3685	29.73	0.8885	1.30	1.81	single Nvidia Geforce GTX 1070
Our 2018 Approach	0.3486	30.37	0.9206	3.74	2.13	single Nvidia Geforce GTX 1070

When compressing some large images, baseline 2018 and our 2018 approach might raise alarm when GPU memory was not sufficient. This usually would not cause problem if each image was compressed using separate Python session but might cause problem if many images were compressed one after another in the same Python session. Typical problem was that the reconstructed images were incomplete and had meaningless blocks as in. Using a GPU with larger memory or using original pictures with lower resolution should be able to solve this problem.

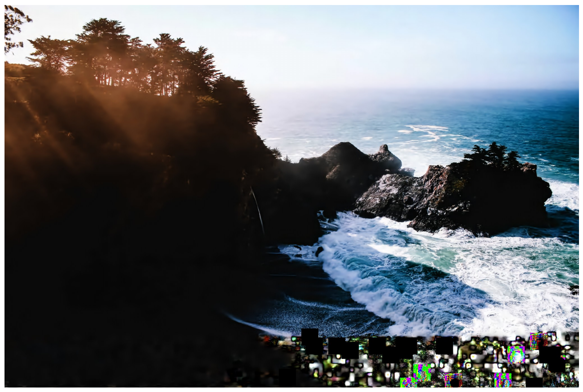

Figure 4.13: An incomplete picture processed by Ballé's approach in 2018. Right-bottom corner was incomplete due to exceeding GPU memory limit and competing with other tasks in the same Python session.

4.2.2 Image Examples

Pictures compressed using both baseline 2018 and our 2018 approach tend to be slightly darker than the original picture when bpp were lower than a specific threshold. We did not explicitly measure this threshold, but we found this phenomenon to be worse when the ambient brightness was relatively low.

One possible reason for this phenomenon is that the hyperprior model did not learn side information very well, and the learned standard deviation was subtracted from the reconstructed image so that the final output looks darker. Another possible reason is that GDN de-correlated two-dimensional densities and caused the picture to lose some ambient light if the density of the ambient light is very similar all over the picture.

However, this phenomenon might become a huge advantage when the original image has white noise caused by over exposure. The de-correlation of two-dimensional densities bring much sharper contrast and richer color details subjectively.

Compared with baseline 2018, our 2018 approach generally achieves around 5% performance improvement with small variance when processing different images.

As is shown in Figure 4.14, the original image of blue ocean is brighter. But in the image compressed by baseline 2018 as is shown in Figure 4.15 and the image compressed by our 2018 approach as is shown in Figure 4.16, the blue water looks much darker, while the foams on brighter parts are easier to see. Comparing with baseline 2018, our 2018 approach achieved around 6.3% reduction in bpp with only 0.63% reduction in PSNR and interestingly 0.05% higher MS-SSIM. The same image compressed by JPEG2000 as in Figure 4.17 has much less PSNR and MS-SSIM at similar bpp.

One interesting observation is that for Figure 4.14, the image compressed by HEIC as in Figure 4.18 actually lost 1 pixel in height, resulting in a resolution of 2048 × 1364 instead of original resolution of 2048 × 1365. This change in resolution was unexpected, but we got the same result after repeating the operation on macOS Mojave 10.14.6.

An example that shows the advantage of the phenomenon is Figure 4.19, which is an original picture with a bit over-exposure. Although the picture compressed by baseline 2018 as in Figure 4.20 and our 2018 approach as in Figure 4.21 is a lit darker, they are at the same time much clearer and natural. For this picture, our 2018 approach achieved around 5.3% reduction of bpp, with around 0.3% reduction in PSNR and MS-SSIM. The image compressed by JPEG2000 as in Figure 4.22 and the image compressed by HEIC as in Figure 4.23 did not look darker but has a much lower PSNR and MS-SSIM at similar bpp, which was visually bad when zoomed in to the original scale.

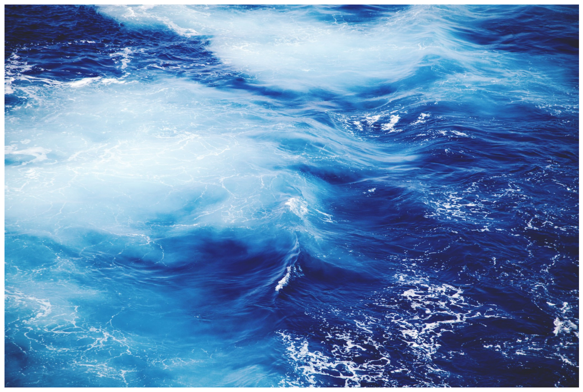

Figure 4.14: Original picture 1 in CLIC professional valid dataset.

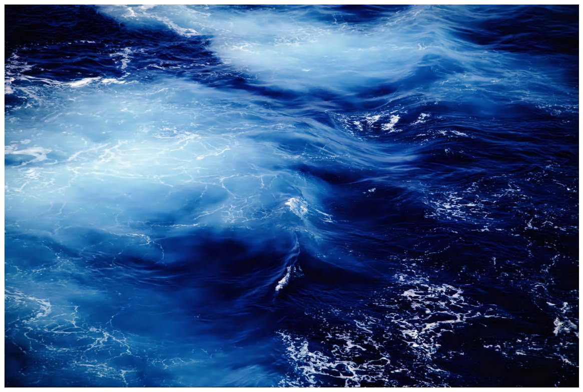

Figure 4.15: Compressed picture 1 by Ballé’s approach in 2018. bpp = 0.3338, PSNR = 32.22 dB, MS-SSIM = 0.9664.

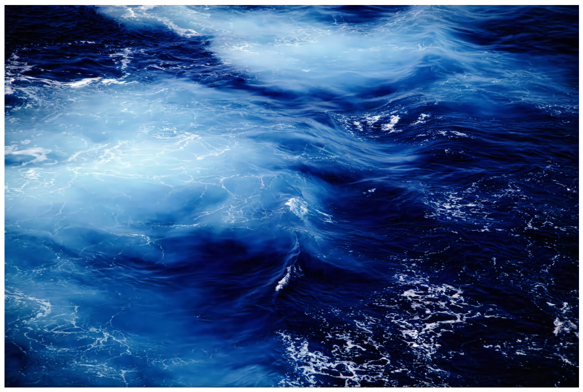

Figure 4.16: Compressed picture 1 by our 2018 approach. bpp = 0.3127, PSNR = 32.01, MS-SSIM = 0.9669.

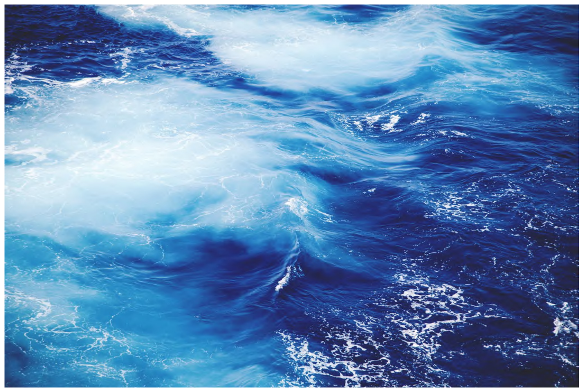

Figure 4.17: Compressed picture 1 by JPEG2000. bpp = 0.3317, PSNR = 13.73, MS-SSIM = 0.8611.

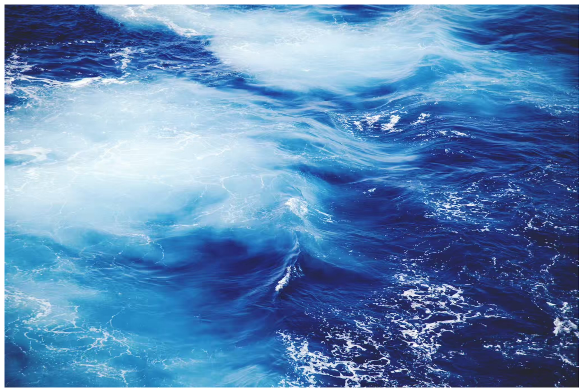

Figure 4.18: Compressed picture 1 by HEIC, lost 1 pixel in height during compression.

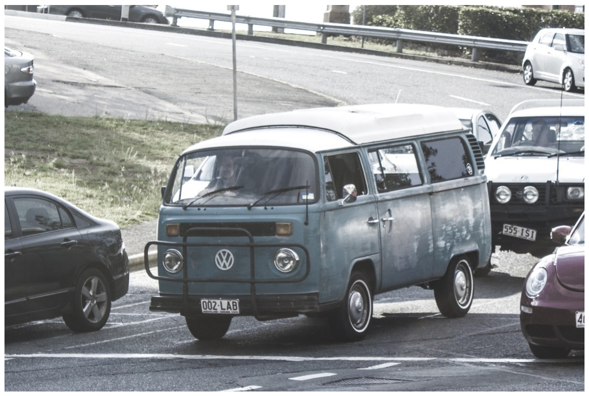

Figure 4.19: Original picture 2 in CLIC professional valid dataset. The picture shows a little over-exposure.

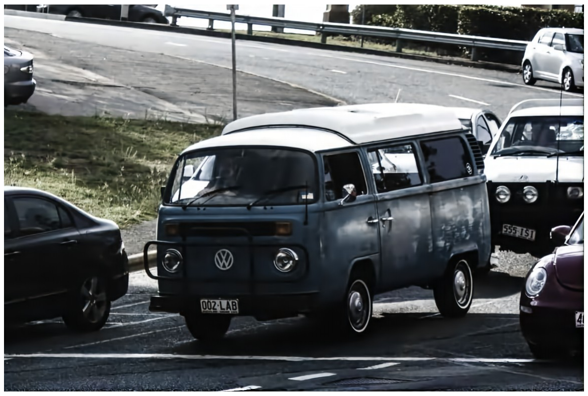

Figure 4.20: Compressed picture 2 by Ballé's approach in 2018. bpp = 0.3603, PSNR = 31.24, MS-SSIM = 0.9789.

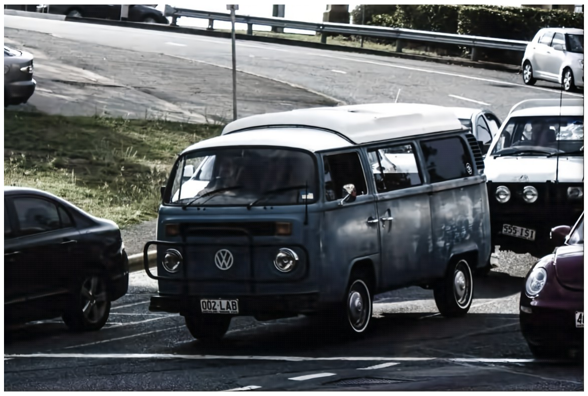

Figure 4.21: Compressed picture 2 by our 2018 approach. bpp = 0.3411, PSNR = 31.11, MS-SSIM = 0.9748.

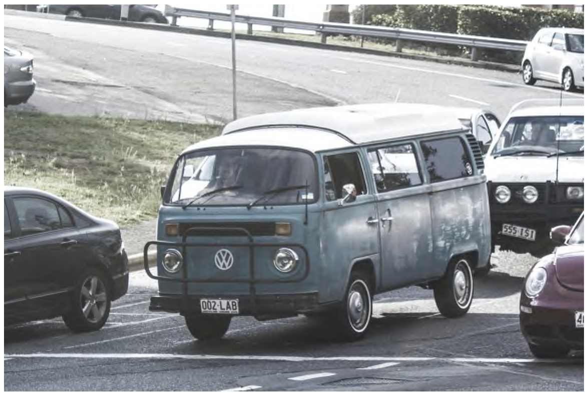

Figure 4.22: Compressed picture 2 by JPEG2000. bpp = 0.3514, PSNR = 12.99, MS-SSIM = 0.8896.

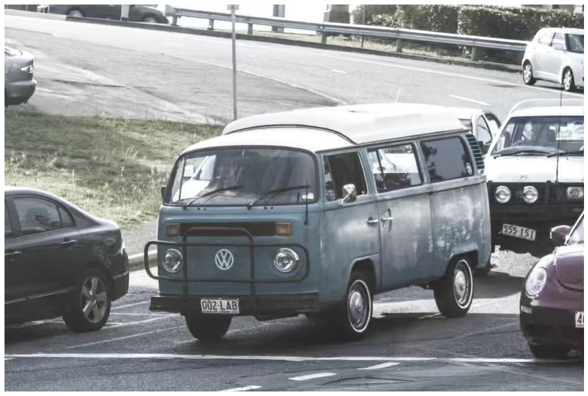

Figure 4.23: Compressed picture 2 by HEIC. bpp = 0.2872, PSNR = 13.00, MS-SSIM = 0.8896.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

In our improvement on Ballé's approach in 2017, we proposed an improved autoencoder with better optimization goal and deeper learned transformer, that outperformed BPG (RGB 4:4:4) in both PSNR and MS-SSIM at comparable bpp. With the same compression ratio, our 2017 approach achieved around 1% improvement on PSNR and 23% improvement on MS-SSIM over BPG (RGB 4:4:4), and slightly outperformed Ballé's approach in 2017.

In our improvement on Ballé's approach in 2018, which is the state-of-the-art image compression approach using deep learning with open source implementation, we proposed an improved autoencoder with deeper learned transformer, that outperformed JPEG2000, HEIC, and Ballé's approach in 2018 in bpp at comparable PSNR and MS-SSIM. Comparing with Ballé's approach in 2018, our 2018 approach achieved around 5.4% reduction in bpp, 2.2% increase in PSNR, and 3.6% increase in MS-SSIM.

In conclusion, we successfully achieved the project goal by improving the state-of-the-art image compression method and achieving the new state-of-the-art results.

5.2 Future Scope

The darker phenomenon in Ballé's approach in 2018 need further investigation and experiment. This issue must be well-solved before this deep learning approach becomes ready for industrial application. The resolution change problem of HEIC is also an interesting bug discovered during experiments.

We think this is not acceptable for image compression codecs, and sincerely hope the expert groups can resolve this in the future.

We truly believe that deep learning based approaches still have lots of potentials in image and video compression, and we plan to dedicate more time in improving our 2018 approach further in the future.

Bibliography

[1] Rafael C Gonzales and Richard E Woods. Digital Image Processing, 2017.

[2] Ookla. Speedtest Global Index, 2019. https://www.speedtest.net/global-index.

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston.,"Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[4] Johannes Ballé, Sung Jin Hwang, and Nick Johnston. Data compression in TensorFlow, 2018. https://github.com/tensorflow/compression.

[5] Licheng Xiao, Hairong Wang, and Nam Ling, "Image Compression with Deeper Learned Transformer," *Proceedings of the APSIPA Annual Summit and Conference 2019*, pp. 53-57, Lanzhou, China, Nov 18-21, 2019.

[6] Descampe Antonin. OpenJPEG 2.3.0, 2017. https://www.openjpeg.org/.

[7] Fabrice Bellard. BPG Image format, 2018. https://bellard.org/bpg/.

[8] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool, "Generative Adversarial Networks for Extreme Learned Image Compression," *Proceedings of the IEEE International Conference on Computer Vision*, pages 221–231, 2019.

[9] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 136–144, 2017.

[10] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, "End-to-end Optimized Image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[11] David Minnen, Johannes Ballé, and George D Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

[12] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack, “Context-Adaptive Entropy Model for End-to-end Optimized Image Compression,” *arXiv preprint arXiv:1809.10452*, 2018.

[13] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, “Density Modeling of Images using a Generalized Normalization Transformation,” *arXiv preprint arXiv:1511.06281*, 2015.

[14] Alex Krizhevsky, Geoffrey Hinton, et al., “Learning Multiple Layers of Features from Tiny Images, Vol. 1. No. 4 Technical report, University of Toronto 2009.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” *arXiv preprint arXiv:1710.10196*, 2017.

[16] Google. CLIC “Workshop and challenge on learned image compression”, 2018. https://www.compression.cc.

[17] Kodak lossless true color image suite. http://r0k.us/graphics/kodak.