Learned Image Compression based on Recurrent Neural Networks with Hyperpriors

ハイパープライアーを用いたリカレントニューラルネットワークに基づいた学習画像圧縮

A Thesis Submitted to the Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering of Waseda University
in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: January 29th, 2020

Rige Su
(5118FG06-4)
Advisor: Prof. Jiro Katto
Research guidance: Research on Image Information

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Image compression

Humans usually use the eyes to observe and recognize the world. Related research shows that external information that human vision receives accounts for 83% of senses. In addition, being the direct projection of the real world in human eyes, images can also be recorded and transmitted in real media by being stored on some mediums. From the early manual painting, to the later film, to the digital images stored in electronic devices in modern society, the storage media of the images are constantly improved and updated, and we can record the world more and more accurately.

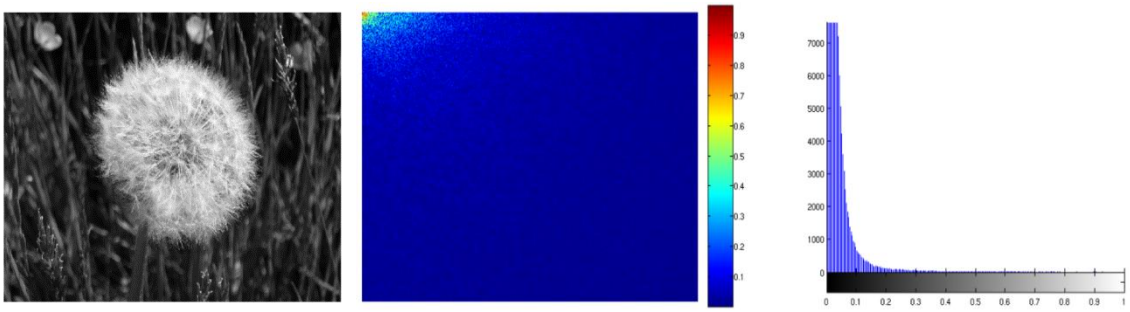
Firstly, we will introduce the basic process of image compression. The image compression consists encoder and decoder, where encoder compresses the uncompressed original images, and generates the bitstream for storage and transmission. The decoder reconstructs the bitstream into images. We mainly introduce the basic concepts and principles in image compression. The core of image compression is to reduce redundant information in the data, so the process of image compression is to make full use of spatial correlation to reduce redundant information and reduce the storage space of the image.

1.1.1 Spatial Correlation

The information of the adjacent points of the image in the two-dimensional space has a strong correlation. Therefore, before doing quantization and entropy coding, we need to consider how to fully reduce the redundancy of these spatial information. Its methods include discrete cosine transform (DCT)[1], discrete wavelet transform, and mode prediction commonly used in Intra coding of video coding for energy concentration. Take DCT as an example, as shown in the Fig.1, the left image is transformed by DCT, the coefficient matrix shown in the middle. The DCT has the important information of the image concentrated on a few coefficients of the transformation, as shown in the right

spectrogram, mainly concentrated in the upper left corner. Therefore, when doing entropy coding, the bitrate can be greatly reduced, which is conducive to image compression.

Fig. 1: Discrete cosine transform and energy distribution



1.1.2 Quantization

After transforming the image, quantization can store floating point numbers with less space with losing a little of the accuracy. Different from lossless compression, quantization is the source of the loss of image accuracy in the compression algorithm. Quantization transforms the transformed image from floating-point numbers to fixed-point numbers, to be used as input for subsequent entropy coding. Common quantization methods include scalar quantization and vector quantization. Scalar quantization is a very common quantization method. Corresponding to vector quantization, it deals with one-dimensional values. By dividing the limited range of the one-dimensional space into several sub-intervals according to certain rules, scalar quantization represents each value to be quantified by the representative value of its sub-interval, thereby reducing the number of bits occupied by data.

1.1.3 Entropy Coding

In the quantized original uncompressed data, each value is represented by a fixed-length bit, and the occurrence probability of different values in the actual value is not completely uniform. Characters with a high probability of occurrence are allocated a binary code with a short character length, and characters with a low probability of occurrence are allocated a binary code with a long character length, so that the average encoding length of a character is the shortest. This can further improve the storage efficiency of the data while preserving entropy (without losing any information). This method of encoding according to the probability distribution characteristics of the median value of the data is called entropy coding. Entropy coding is lossless coding.

Commonly used entropy coding methods include Huffman coding[3], arithmetic coding[5], and CAVLC[6] and CABAC[13] used in video coding. Taking the Huffman coding as an example, Huffman coding is a form of statistical coding that attempts to reduce the bits number, which required to represent a symbol string. Huffman tree, also known as optimal binary tree, is a binary tree with the shortest weighted length of path. The so-called weighted path length of a tree is the weight of all leaf nodes in the tree multiplied by the path length to the root node (if the root node is 0 layers, the length of path from the leaf node to the root node is the number of layers of the node).

Huffman encoding is a very effective encoding method widely used for data file compression. Its compression ratio is usually between 20% and 90%. Suppose there is a file containing 100,000 characters, and each character appears differently, as shown in the Table 1.

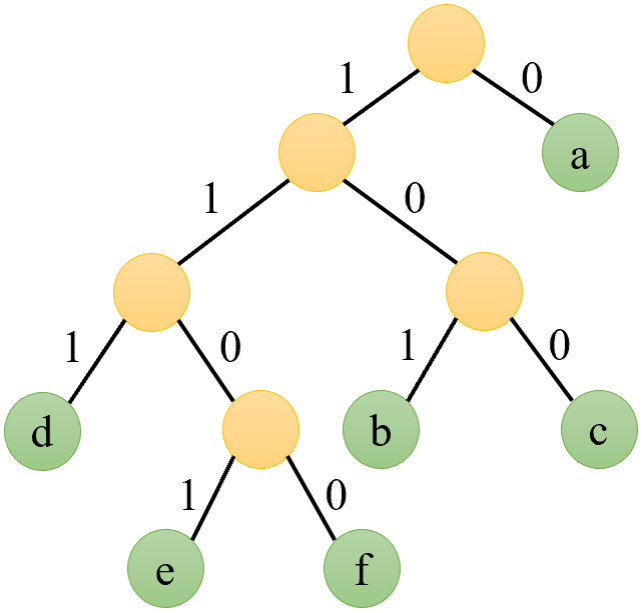
Symbols	a	b	c	d	e	f
Frequency (10^3)	40	10	9	13	6	2
Fixed code length	000	001	010	011	100	101
Fixed variable length	0	101	100	111	1101	1100

Table 1: Huffman code table

There are multiple ways to represent the information in the file. If you use the 0,1 code to represent characters, that is, each character is represented by a unique 0,1 string. If the fixed-length encoding is used, 3 characters are required to represent one character, and the entire file encoding is required to be 240,000 bits. If the variable-length encoding is used as shown in Fig. 2, calculate the average code length by the following formula:

$$L_{average} = \sum_{i=0}^{N} p_i L_i \tag{1}$$

Fig. 2: Huffman tree



where, L is the code length. We can get that $(40 \times 1 + 10 \times 3 + 9 \times 3 + 13 \times 3 + 6 \times 4 + 2 \times 4) \times 1000 = 168{,}000$ bits. The variable length code is better than the fixed length code scheme, and the total code length is reduced by about 30%.

1.2 Evaluation of Image Compression Quality

There are two important indicators for evaluating the quality of a compression algorithm, which are bit per pixel, and reconstruction quality. The bit rate represents the size of the storage space occupied by the compressed image. The formula is as follows,

$$bpp = \frac{BitrateSize}{ImageHeight \times ImageWidth} \tag{2}$$

Image compression technology has been commonly utilized in various fields, and the evaluation of images quality has become a basic subject. Because image information has incomparable advantages over other information, reasonable processing of image information has become an indispensable means in various fields. For example, medical images have very high requirements for image quality because they will affect the judgment of the disease. Systems such as teleconferencing and video playback are affected by unfavorable factors such as transmission errors and network delays and need to dynamically adjust the image quality to adapt to changing network conditions; for computer vision, different tasks have very different tolerances for image distortion. On the

other hand, the focus of the machine on the image and the human eyes are also very different. Therefore, reasonable image quality evaluation indexes have important application value.

1.2.1 PSNR

Peak signal to noise ratio (PSNR)[8], a full reference image quality evaluation indicator. The formula of PSNR as follow,

$$MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(X(i,j) - Y(i,j) \right)^2 \tag{3}$$

$$PSNR = 10 log_{10} \left(\frac{(2^n - 1)^2}{MSE} \right) \tag{4}$$

From the formula that the MSE represents the mean square error of the input image X and the reference image Y. In addition, H and W are the height and width of the image, respectively; n is the number of bits per pixel, it always set as 8, that is, the number of pixel gray levels is 256. The unit of PSNR is dB. The smaller the MSE, the larger the PSNR, which means that the closer the reconstructed image is to the original image, the smaller the image loss. PSNR is the earliest objective evaluation index of images quality. From the research results and experiments in recent years, it can be found that the PSNR cannot fully express the subjective visual perception of the human eyes, and sometimes the subjective quality of the reconstructed image with a high PSNR value is worse than the low PSNR value. Because it is not considered that the human eyes are more sensitive to the contrast difference with lower spatial frequency, and to the contrast difference than the chroma. Besides, the human eye's perception of a region will be affected by its surrounding neighboring areas. Therefore, the evaluation results often do not agree with the subjective feelings of people. Due to the simple calculation of the PSNR and the direct evaluation method, it is still an objective image quality evaluation index widely used at present.

1.2.2 SSIM

Structural similarity index (SSIM)[9] is a measure of the similarity of two images. The difference from traditional image quality indicators (such as MSE or PSNR) is that these original methods are estimating absolute errors, while SSIM is a perception-based model

that treats image degradation as a perceived change. At the same time, important visual phenomena such as brightness masking and contrast masking are also integrated. The idea of structural information comes from the strong interdependence between pixels, especially when they are close in space, these dependencies includes important information about the structure in the visual scene. Brightness occlusion is a phenomenon in which image distortion is more difficult to find in bright areas. Contrast occlusion is a phenomenon in which distortion becomes less noticeable when there is a significant texture or the like in an image.

As a concrete realization of this kind of theory, in SSIM, the mean value is used to estimate the brightness, the standard deviation is used to estimate the contrast, the covariance value is used to estimate the structural similarity, and the three are combined into the final value of the SSIM, which shown as follow,

$$l(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1} \tag{5}$$

$$c(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2} \tag{6}$$

$$s(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3} \tag{7}$$

where, $\mu_X,\ \mu_Y$ are represent the means of X and Y, respectively. $\sigma_X,\ \sigma_Y$ are represent the variance of X and Y, respectively. $\sigma_X\sigma_Y$ is the covariance. $C_1,\ C_2,\ C_3$ are the constant, and to avoid the denominator being zero, set $C_1 = (k_1 * L)^2$, $C_2 = (k_2 * L)^2$, $C_3 = C_2/2$, where default the $k_1 = 0.01$, $k_2 = 0.03$, $L = 255$, so the SSIM can be shown as follow,

$$SSIM(X,Y) = l(X,Y) \cdot c(X,Y) \cdot s(X,Y) \tag{8}$$

where, the range of SSIM is [0,1], so larger the value, the smaller the image distortion.

1.2.3 MS-SSIM

MS-SSIM[10] is an improvement on SSIM. MS is the abbreviation of Multi-Scale. It performs calculations on multiple scales through multiple stages of down sampling. Many

experiments have shown that MS-SSIM is like or better than SSIM in measuring subjective quality. The formula of MS-SSIM as follow,

$$MS-SSIM(X,Y) = [l_M(X,Y)]^{\alpha M} \cdot \prod_{j=1}^{M} [c_j(X,Y)]^{\beta_j} [s_j(X,Y)]^{\gamma_j} \quad (9)$$

where, the scale of the original image is 1, and the highest scale is M. αM, β_j, γ_j adjust the proportion of each part.

1.3 Main Contribution

This thesis main has three contributions which can be listed as follows:

First, image compression has become a fundamental research topic. This thesis is to study image compression based on deep learning. Recently learned Image compression has achieved many great progresses which benefit from fast development of deep learning. Currently, convolutional neural networks (CNNs) are widely used in majority of learned image compression approaches and achieved great success. However, CNNs are not fit for scalable coding and multiple models need to be trained separately to achieve different rates. Therefore, we propose a scalable learned image compression architecture based on recurrent neural networks (RNNs). Different from existing RNN-based image compression methods, we present an RNN architecture with quantization to make the feature maps of RNNs more compressible. Then, we utilize the entropy coding to further reduce the redundancy and generate the bitstream.

Second, in order to realize the scalable coding, we allocate the bits to multiple layers, by adjusting the layer-wise lambda values in Lagrangian multiplier-based rate-distortion optimization function. Experimental results demonstrate that our performance can be comparable with traditional image coding algorithms and existing RNN-based methods on Kodak dataset. Besides, our method is scalable and flexible coding approach, to achieve multiple rates using one single model, which is very appealing in practice.

Third, in order to further optimize the bitrate, this thesis first tried to adopt RNN-based hyperprior to multiple layers.

1.4 Outline

The rest of this thesis is organized as follows. Chapter 2 discusses the related work of image compression. Chapter 3 describes the method we proposed in this thesis. We show our experiment results in chapter 4, and in this chapter, we also analysis our experiment results. In the chapter 5, we discuss our research and summarize the thesis and propose the future work. In the chapter 6, we do the conclusions of our research work.

Chapter 2

Related Work

2.1 Traditional Image Compression

2.1.1 JPEG

Joint photographic exports group (JPEG)[1] is a lossy compression standard method widely used for images. It is the first international image compression standard. As the earliest generation of image compression standards, JPEG is difficult to compare with subsequent improved versions in performance, but for various reasons, it is still the most widely used image compression algorithm. JPEG has very good reconstruction quality in the middle and high bit rate segments, but in the low bit rate part, its quality will quickly decline, and obvious block effects will appear.

Taking the benchmark JPEG algorithm to compress a 24-bit color image as an example, the compression steps are as follows.

a. Color mode conversion

JPEG uses the YC_bC_r color space, while BMP uses the RGB color space. If you want to compress a BMP image, you will first convert the RGB format image to the YC_bC_r (YUV) format. YUV is also a common color-coding method. RGB reflects the perception of color by the human eye. YUV highlights the sensitivity of the human eye to brightness. Its application areas include television systems and analog video. Among them, Y represents the luminance channel, and U and V represent the chroma and saturation respectively. If the U and V channels are ignored, the separate Y channel can also be displayed on a black and white TV, so that the black and white TV and the color TV are compatible in signal. When using YUV format to save images, human visual perception characteristics can be used to reduce the chroma channel bandwidth moderately.

The conversion formula[1] between RGB and $YC_bC_r$is as follows,

$$Y = 0.299 * \mathrm{R} + 0.587 * \mathrm{G} + 0.114 * \mathrm{B} \quad (10)$$

$$C_b = -0.1687 * \mathrm{R} - 0.3313 * \mathrm{G} + 0.5 * \mathrm{B} + 128 \quad (11)$$

$$C_r = 0.5 * \mathrm{R} - 0.4187 * \mathrm{G} - 0.0813 * \mathrm{B} + 128 \quad (12)$$

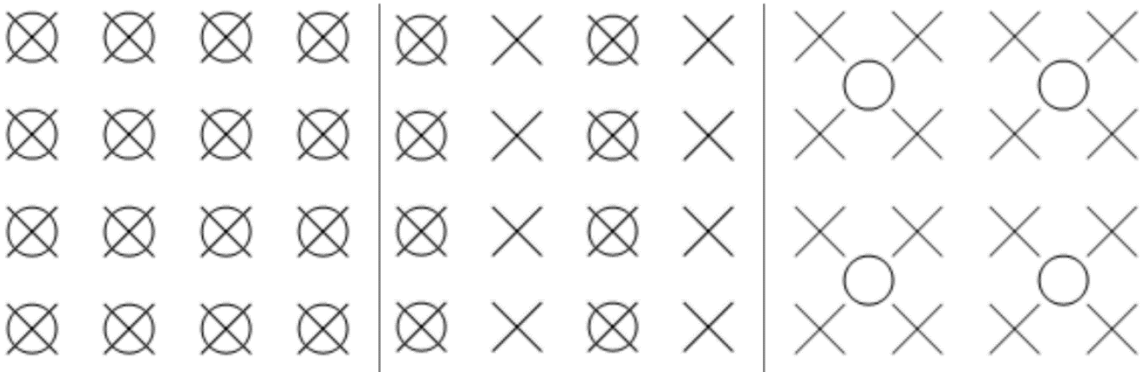
b. DC level offset

The effect of DC offset is to make the dynamic range of the input image approximately concentrated around 0. Taking 8bits data as an example, the original range of the data is [0, 255]. In order to reduce the absolute value fluctuation, first shift the value (minus 128) to [-128, 127].

c. Sampling

Studies have shown that the human eyes are much more sensitive to brightness transformations than to color transformations. Therefore, we can think that the Y component is more important than the UV component, so we can down-sample the UV channel to reduce the data storage as much as possible without the loss of visual quality. YUV has many different sampling methods, the common ones include YUV4: 4: 4, YUV4: 2: 2 and YUV4: 2: 0[4]. The Fig.3 shows the different sampling methods,

Fig. 3: From left to right are YUV4: 4: 4, YUV4: 2: 2 and YUV4: 2: 0, respectively



where, the cross represents the Y component, the circle represents the UV component, and the amount of sampled data decreases from left to right. YUV4: 2: 0 is very widely used in video and image encoding tasks. Although this sampling method loses a certain degree of accuracy, it reduces the amount of data stored in a range that the human eye does not perceive.

d. Block

Because DCT processes 8x8 sub-blocks, the original image data must be divided into blocks before performing DCT. Because the three components of each point in the original image appear alternately, the three components must first be separated and stored in three tables. After reading 8x8 sub-blocks from left to right and from top to bottom, and storing them in a table of length 64, DCT can be performed.

JPEG is processed every 8x8 points as a unit. So, if the length and width of the original image is not a multiple of 8, you need to fill in multiples of 8 first so that it can be processed block by block. After the original image data is divided into a matrix of 8 * 8 data units, each value must be subtracted from 128 and then brought into the DCT transformation formula one by one to achieve the purpose of DCT. The data value of the image must be subtracted from 128 because the numerical range accepted by the DCT formula is between -128 and 127.

e. DCT

Discrete cosine transform (DCT) is a transform coding method commonly used in bit rate compression. It has the function of separating high and low frequency information, so after the image block passes DCT, the DC component and low frequency signal will be concentrated in the upper left corner of the block. The higher frequency information will be in the lower right corner. Based on this feature, you can design corresponding quantization and coding strategies to further save the bit rate. After dividing the image into multiple 8x8 matrices, and then performing DCT on each pixel block, the image is transformed into a matrix of frequency coefficients composed of floating-point numbers.

The formula for forward DCT is as follows,

$$F_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right] \qquad (13)$$

where, $0 \le u, v < 8$, $g_{x,y}$ is the pixel block.

$$\alpha(u), \alpha(v) = \begin{cases} \frac{1}{\sqrt{2}}, & if\ u, v = 0 \\ 1, & else \end{cases} \qquad (14)$$

In decoding, the inverse discrete cosine transform (IDCT) is used to restore the frequency domain information to pixel space,

$$f_{x,y} = \frac{1}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}\alpha(u)\alpha(v)F_{u,v}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right] \quad (15)$$

f. Quantization

DCT has realized the separation of high and low frequency signals. At this time, the large value in the 8x8 block is concentrated in the upper left corner. JPEG determines the quantization level at different positions through the quality factor. The lower the value of the quality factor (1 is the minimum value), the higher the image reconstruction quality and the lower the compression ratio. Conversely, the higher the value, the worse the image reconstruction quality, but the higher the compression ratio.

ISO specifically developed a standard quantization table for JPEG, as shown in the following table. These two watches are designed based on psychological vision experiments performed by Lohscheller. These quantization tables can have relatively good results on most natural images with 8bits accuracy. The quantization table is the key to controlling the JPEG compression ratio. This step removes some high frequency amounts and loses a lot of detailed information and cannot be applied to all images.

Standard brightness quantization table,

$$Q_Y = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 99 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 99 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 99 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (16)$$

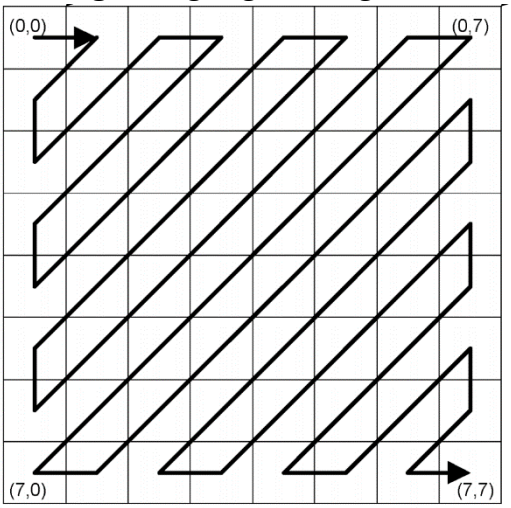
Standard color quantization table,

$$Q_C = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \quad (17)$$

g. Entropy coding

First, zig-zag[11] ordering is used for encoding, shown in Fig.4. This is because in the 8x8 matrix, many values are concentrated in the upper left corner, and many zero values exist in the lower right corner. The zigzag sequence can maximize the function of run-length encoding. Second, run-length encoding (RLE) is used to encode the AC coefficient. The zero-valued RLE is for consecutive zero values. Instead of repeatedly storing each zero, it records the number of consecutive occurrences of zero. To achieve the purpose of reducing the amount of data. Third, the differential pulse code modulation (DPCM) is used to encode the direct current coefficient (DC). Finally, entropy coding is used for coding. Huffman coding or arithmetic coding is allowed in JPEG.

Fig. 4: Zig-zag ordering[11]



2.1.2 JPEG2000

JPEG2000 supports progressive encoding, transmission, and lossy and lossless compression[2]. Compared with JPEG, JPEG2000 has obvious advantages in the case of high compression rate. At higher compression ratios (more than 100 times), JPEG2000 can still maintain a good reconstruction quality, but JPEG has been severely distorted and

cannot be used. In the case of low compression rate (less than 10 times), the traditional JPEG image quality may be better than JPEG2000. The distortion of JPEG2000 is mainly fuzzy distortion, which is mainly caused by the loss of high-frequency information. Of course, this problem also exists in the traditional JPEG standard. JPEG2000 can achieve progressive transmission, which is an extremely important feature of JPEG2000. This is what we often call the "fade-in" feature of GIF-format images. It first transfers the outline of the image, then gradually transfers the data, and continuously improves the image quality, so that the image is displayed from hazy to clear, instead of slowly displaying from top to bottom like the current JPEG. So, in general, JPEG2000 is a new compression method that is much better than traditional JPEG. Under the same reconstruction quality, JPEG2000 can usually save about 30% of storage space.

The core content of JPEG2000 has the following points,

a. Discrete wavelet transform

JEPG2000 replaces the traditional DCT transform of JPEG with discrete wavelet transform (DWT), which can further reduce the correlation between data. Compared with the DCT used by JPEG, DWT has good locality, can use different spatial frequency resolutions for different regions in different types of images, so it is possible to achieve better compression ratios, and it can also achieve lossless compression. For example, one-dimensional DWT is a series of high-pass and low-pass filtering on the source signal and reduces the data sampling frequency to half of the original after time conversion to ensure that the coefficients obtained after each wavelet transform are the same as the number of source signals. Each time the low-pass filtered output saves the low-frequency information of the source information. It is a reproduction of the source signal at a lower resolution, which concentrates most of the energy in the source signal. The high-pass filtered output saves the source signal High-frequency information, such as boundaries and materials, contains very little energy. A low pass filtered signal still has a lot of correlation. To improve the compression performance, it still needs to be filtered again until the correlation between the signals reaches a negligible degree. Because the high pass filtered signal has very small energy, it is often not cost effective to filter it, so it is generally no longer filtered. This filtering method is called dyadic decomposition.

b. EBCOT:

Embedded block coding with optimized truncation (EBCOT) is a coding algorithm published by David Taubman in 1998[12]. EBCOT is the core of the JPEG2000 standard. It is an embedded bit-layer coding method of wavelet coefficients. It mainly includes embedded bit-layer coding, optimized truncated bit stream ordering, and Multiple Quantization (MQ) arithmetic entropy coding. MQ coding is an adaptive binary arithmetic coding. It can be divided into two parts, tier1 and tier2.

The first is Tier, which divides each sub band into independent coding blocks and decomposes the wavelet coefficients in each block into a bit plane. Encoding is coded on a scan-by-scan basis from the non-zero most significant bit plane (MSB plane) to the lowest bit plane (LSB plane) of the coded block. Then, each coded block is subjected to embedded code scanning independently. Three scans are performed on each bit plane, and the bits on the bit plane are divided into three different coding channels according to certain rules. Next, the results of the three scanning processes are sequentially subjected to conditional encoding of the bit plane. The encoded bitstream and corresponding context information generated after the conditional encoding are sent to an MQ arithmetic encoder for encoding to obtain an embedded bitstream.

The second is Tier2. The output of Tier1 is a series of encoded channels. In the case of lossy encoding, you can weigh the bit rate and reconstruction quality according to the output code rate requirements. By combining the embedded code streams of each encoding block, find the most optimal truncation point. According to this truncation point, only the most important coding channels in the code stream are retained, and other channels are discarded. This is another major source of information loss for JPEG2000 beyond quantization. A series of encoded channel information that is finally retained will be packed and then output into a JPEG2000 code stream.

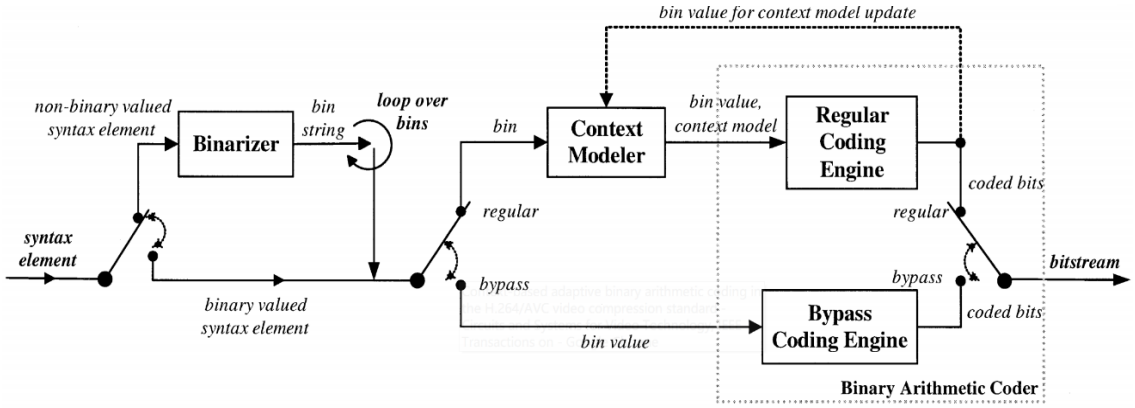
2.1.3 BPG

Better Portable Graphics (BPG) is a new image format. The main features of BPG are that it is based on the HEVC video compression standard, has a higher compression rate under the same recovery quality, and supports lossless compression. Since BPG is based

on the HEVC standard, the HEVC is the best performing image compression algorithm, currently.

HEVC uses Context-based Adaptive Binary Arithmetic Coding (CABAC)[13] as the lossless entropy coding module. The specific implementation of CABAC is introduced into four parts as follow, the CABAC flow diagram is shown in Fig.5.

Fig. 5: Context-based Adaptive Binary Arithmetic Coding [13]



a. Binarization

CABAC uses binary arithmetic coding, which means that only two digits 1 or 0 can be encoded. A non-binary numeric symbol, such as a conversion coefficient, is binarized or converted into a binary codeword before arithmetic coding. This binary codeword is further encoded by an arithmetic encoder before transmission.

b. Context model selection

The context model is a probability model. This model is a model selected based on the statistics of the most recently encoded data symbol. This model holds the probability that each bin is 1 or 0.

c. Arithmetic coding

Arithmetic coding is a lossless data compression method and an entropy coding method. Other entropy coding methods usually divide the input message into symbols and then encode each symbol. Different from other entropy coding methods, the arithmetic coding

directly encodes the entire input message into a number. Here The arithmetic encoder encodes each bin according to the selected probability model.

d. Probability update

The context model is updated based on the actual encoding value. For example, if the value of bin is 1, the frequency count of 1 is incremented.

Value	Binarization				
0	0				
1	1	0			
2	1	1	0		
3	1	1	1	0	
4	1	1	1	1	0

Table 2: Binarization table

Before CABAC performs context modeling and arithmetic coding, it needs to convert the data to be coded into a binary code stream that meets the requirements of binary arithmetic coding according to certain rules. This process is called binarization. Taking the simplest unary code as an example, for a non-binary unsigned integer value symbol n, a unary code can be composed of n ones and a trailing 0, as shown in the table below. For example, the input un-coded value is 4, and the result of binarizing the unary code is 11110.

2.2 Deep Learning for Image Compression

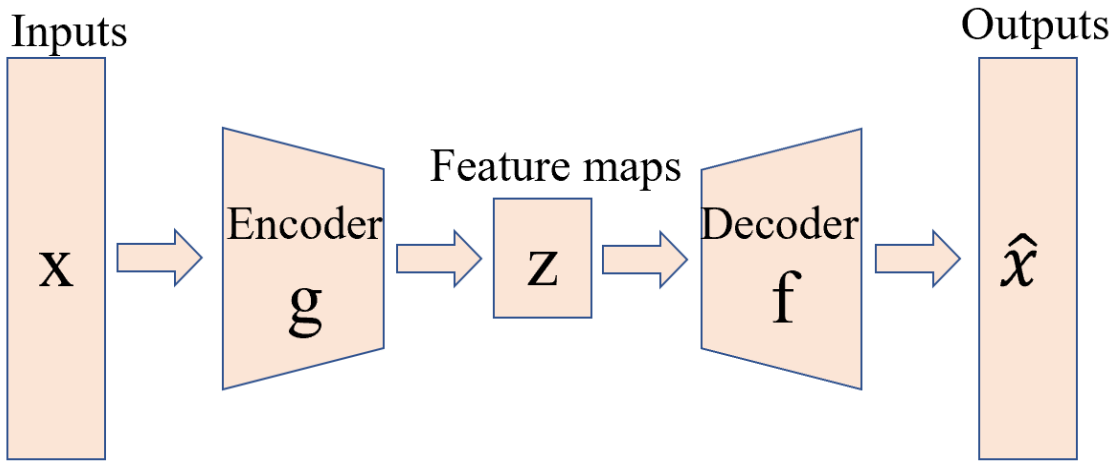
2.2.1 Convolutional Neural Network

a. Autoencoder

The Autoencoder[14] framework contains two major modules: the encoding and decoding process. The procedure of autoencoder shown in Fig.6. The input sample x is mapped to the feature space z through encode, which is the encoding process; then the abstract feature z is mapped back to the original space through decode to obtain the reconstructed sample $\hat{x}$, which is the decoding process. The optimization goal is to

optimize the encoder and decoder at the same time by minimizing the reconstruction error, thereby learning to obtain the abstract feature representation z for the sample input x.

Fig. 6: Image compression progressing of autoencoder



Autoencoder does not need to use sample labels in the optimization process. Essentially, the input of the sample is used as the input and output of the neural network at the same time. By minimizing the reconstruction error, the abstract feature representation z of the sample can be learned. This unsupervised optimization method greatly improves the generality of the model.

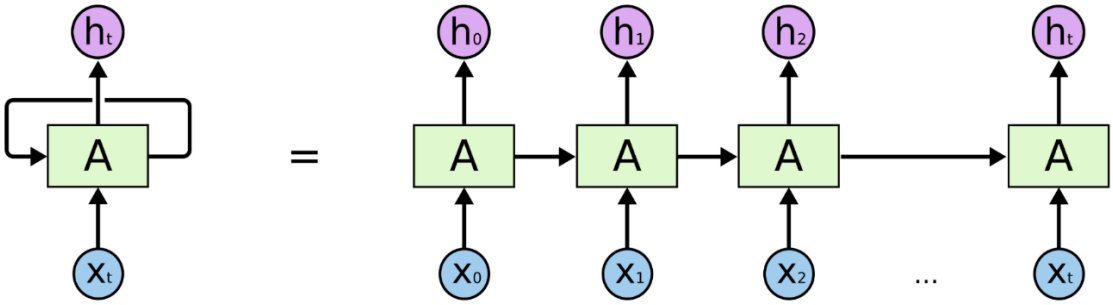
For the Autoencoder model based on neural network, the encoder part compresses the data by reducing the number of neurons layer by layer, and the decoder part increases the number of neurons layer by layer based on the abstract representation of the data, and finally realizes the reconstruction of the input samples.

2.2.2 Recurrent Neural Network

The Fig.7 shows the recurrent neural network (RNN) structure[15]. After expansion, a sequence structure will be obtained. The previous output will be used as the next input (that is, the previous output will affect the subsequent input). This chain-like feature reveals that the RNN is essentially related to sequences, so this sequence is very suitable for processing speech, text, and so on. The key point of RNN is the ability to connect previous information to the current task, such as inferring the meaning of the current sentence through the previous text. However, when the distance between the relevant information and the

current sentence is too large, it will be difficult for the RNN to learn long-distance information. Therefore, the following RNN network structure is derived.

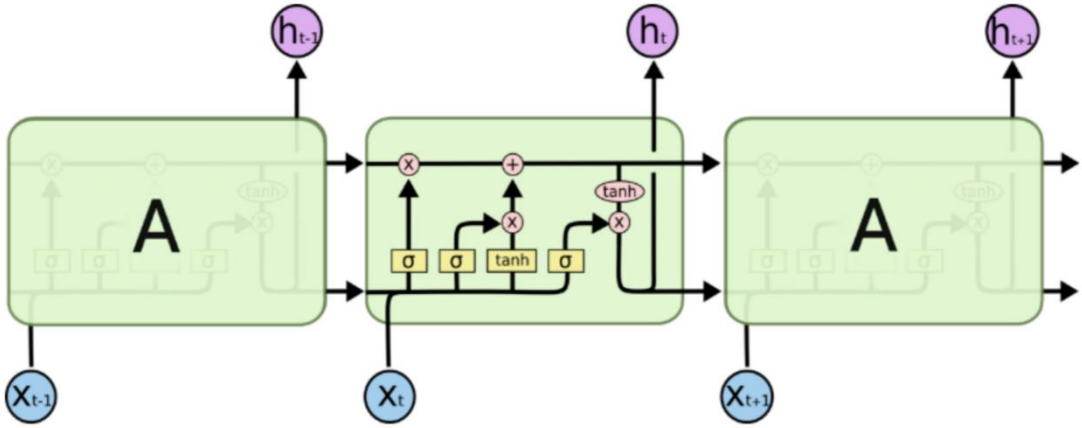
Fig. 7: Recurrent neural network architecture



a. LSTM

Long short-term memory (LSTM)[16] is a special type of RNN, the architecture of LSTM is shown in Fig.8. Through exquisite design, the problem of gradient disappearance and gradient explosion during long sequence training can be solved, that is, the problem of information loss caused by long distance transmission is solved.

Fig. 8: The LSTM architecture of three units



The core of LSTM is the cell state, that is, the horizontal line running from left to right above the LSTM cell in the figure. It is like a conveyor belt, passing information from the previous unit to the next unit, and there is only a small linear interaction with other parts. LSTM uses gate to control discard or increase information, to realize the function of forgetting or remembering. A gate is a structure that allows information to pass selectively and consists of a sigmoid function and a dot multiplication operation. The output value of

the sigmoid function is in the interval [0,1], where 0 means completely discarded and 1 means completely passed. An LSTM unit has three gates, the forget gate, the input gate, and the output gate, respectively.

Forget gate: The forget gate is a sigmoid function where the output h_{t-1} of the previous unit and the input x_t of this unit are inputs. Generate a value in [0,1] for each term in C_{t-1}, controlling the degree to which the last cell state is forgotten.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{18}$$

Input gate: The input gate works with a $tanh$ function to control what new information is added. The tanh function generates a new candidate vector $\widetilde{C_t}$, and the input gate generates a value in [0,1] for each of $\widetilde{C_t}$, which controls how much new information is added. At this point, we have the output f_t of the forget gate, which is used to control the degree of forgetting of the previous unit. There is also an output i_t of the input gate, which is used to control how much new information is added, so that the unit status of the memory unit can be updated.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \tag{19}$$

$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i) \tag{20}$$

$$\widetilde{C_t} = \tanh(\mathrm{W_C} \cdot [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_C}) \tag{21}$$

Output gate: The output gate is used to control how much the current unit state is filtered out. The unit state is activated first, and the output gate generates a value within [0,1] for each of them, controlling the degree to which the unit state is filtered.

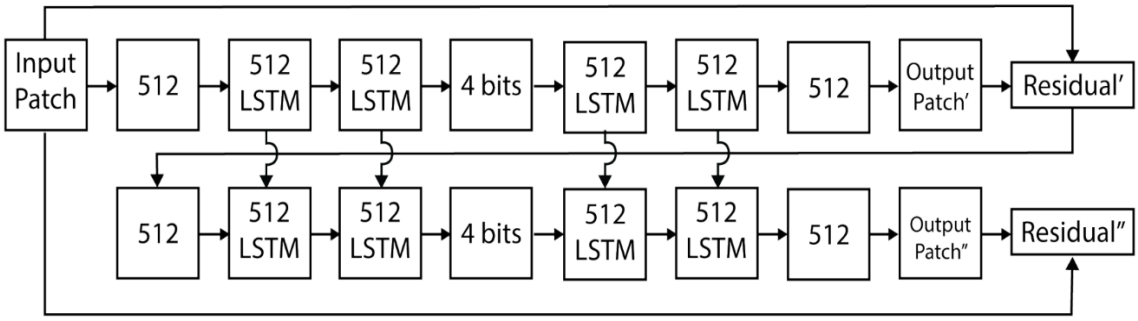
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{22}$$

$$h_t = o_t * \tanh(C_t) \tag{23}$$

b. Produce progressive codes

G.Toderici[19] firstly proposed a LSTM-based image compression architecture. The method that starts to use LSTM and other methods to cyclically output code streams is called produce progressive codes. This method is the same as JPEG2000, which can make the reconstructed image quality better and better with the continuous output of code streams. as shown in Fig.9.

Fig. 9: LSTM-based produce progressive architecture [19]

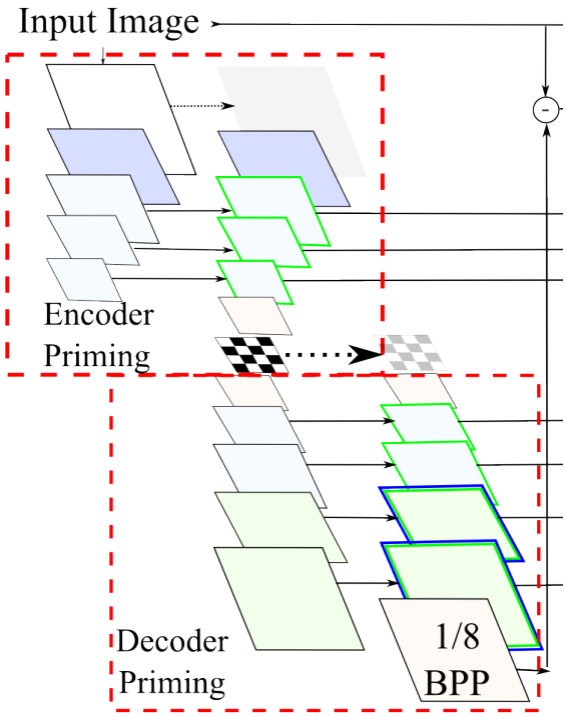


c. Priming

At the first iteration, the hidden state of each GRU layer is initialized to zero. The study[20] found that during the first few iterations, the improvement in image quality was significant. Because the encoder and decoder are stacked with many GRU network layer sequences, the encoder's binarizer and decoder reconstruction require several iterations to observe the improvement of the hidden state of the first layer GRU. Therefore, the research uses hidden-state priming technology to generate a better initial hidden state for each GRU layer.

The hidden state priming, or "k-priming," alone increases the loop depth of the first iteration of the encoder and decoder networks, the network architecture shown in Fig 10. In order to avoid occupying additional bandwidth, these steps are run separately, and the

Fig. 10: The loop depth of the first iteration of encoder-decoder architecture [20]



extra bits generated by the encoder are not added to the actual code stream. For the encoder, this means processing the original image multiple times, discarding the generated bits, but saving changes in the hidden state of the loop unit of the encoder. For the decoder, this means generating decoded images multiple times, but only retaining the final image reconstruction.

Chapter 3

Proposed Method

3.1 Overview

In this paper, we propose a scalable learned image compression architecture based on recurrent neural networks (RNN). First, we present an RNN architecture with quantization to make the feature maps of RNNs more compressible and we utilize the entropy coding to further reduce the redundancy and generate the bitstream. Second, in order to realize the scalable coding, we allocate the bits to multiple layers, by adjusting the layer-wise lambda values in Lagrangian multiplier-based rate-distortion optimization function. Third, we add an RNN-based hyperprior to effectively capture spatial dependencies in potential representations, and further optimize the bit rate.

3.2 Proposed RNN-based Architecture

3.2.1 RNN-based architecture with quantization and entropy coding

When using deep learning to handle image compression and reconstruction, a problem often encountered is that the main battlefield of deep learning has been the field of computer vision for a long time, so a lot of design work is aimed at some tasks such as recognition and detection. When we want to introduce some structures and designs that have been proven to be very effective in the field of computer vision into image compression, they may not produce good results. At this time, we must adjust the structure of the original network or layer.

The basic structure still inherits the autoencoder structure mentioned above. The characteristics of this structure determine that it mainly uses spatial correlation and removes redundant information in space through multiple down sampling. In addition, this paper is based on RNNs, so we use the network structure pattern of produce progressive codes for design.

The single layer of RNN-based image compression architecture with quantization and entropy coder shown in Fig.11. The encoder has two CNNs, three RNNs, a quantization layer and an entropy coder, which we utilized the arithmetic encoder. The input size of the network is $H \times W \times 3$. In general, the forward encoder network uses the first 3×3 CNN layer and three 3×3 RNNs layers for down sampling. Down sampling the image size as $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$, respectively. The decoder has two CNNs, four RNNs, a de-quantization layer and an entropy decoder. Besides, there are also four depth-to-space behind each RNN for up sampling. The architecture of the decoder is a symmetrical structure of the encoder used to reconstruct the signal from the compressed feature maps.

The multiplayer of RNN-based architecture shown in Fig.12. For the first layer, the input is the RGB image passed into encoder-decoder architecture and generate output. The input minus the output get the residual1, then set the residual1 as the input of second layer and generate a residual output named resudual1'. Residual1 minus the residual1' get the residual2, then set the residual2 as the input of third layer, and so on. From the second layer, the input are all the residuals, and generate the residuals as output, so that the final reconstructed image is $output1 + residual1' + residual2' + \cdots + residual\ n'$. For the RNNs have the memory function, so the RNNs of each layer share the weight to each other.

Fig. 11: The single layer of RNN-based image compression architecture with quantization and entropy coding

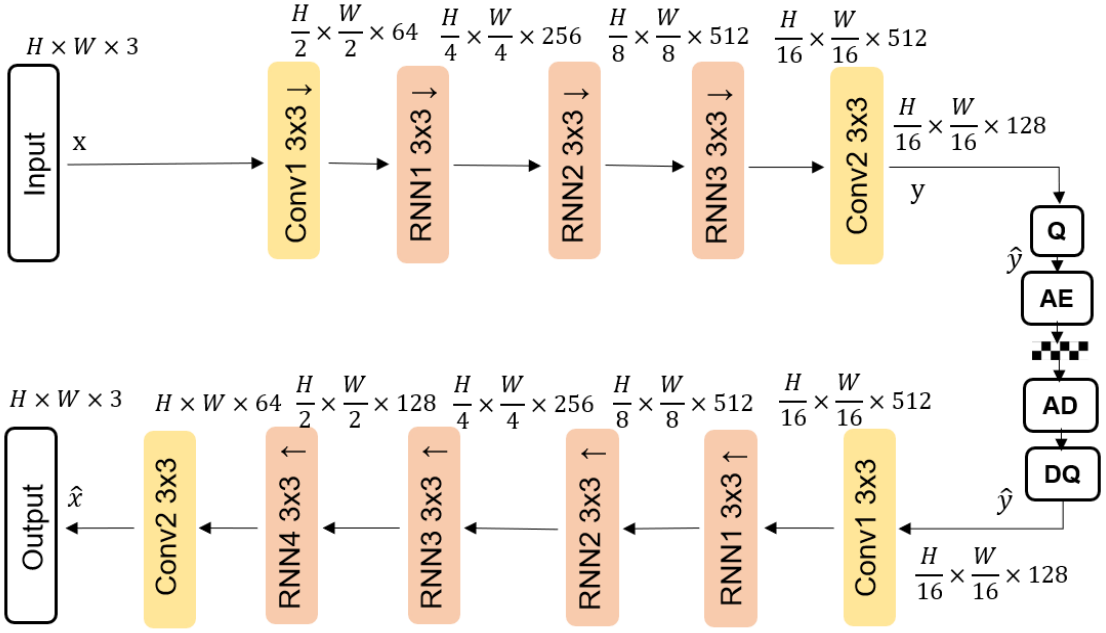
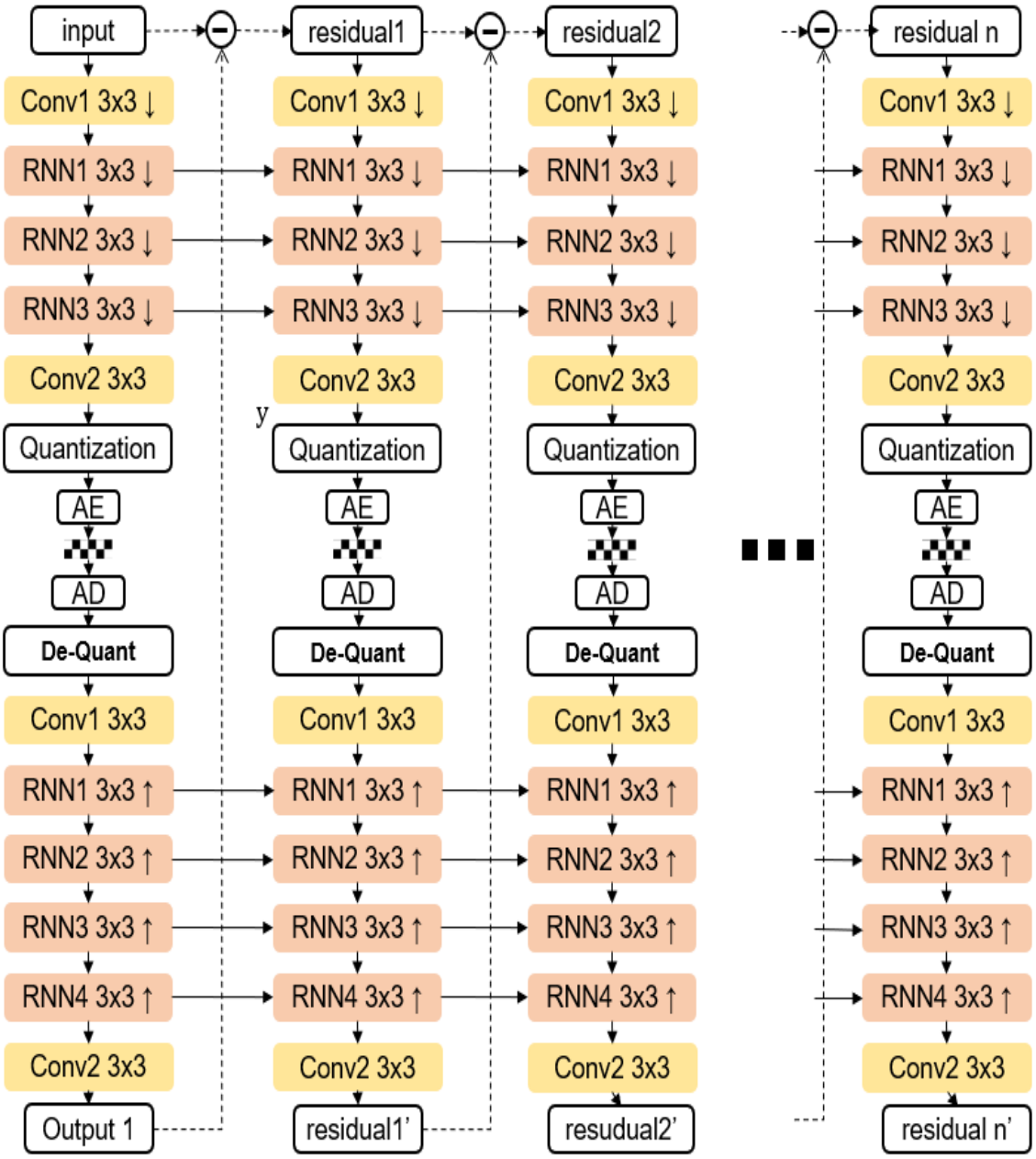


Fig. 12: The multiple layers of RNN-based image compression with quantization and entropy coding

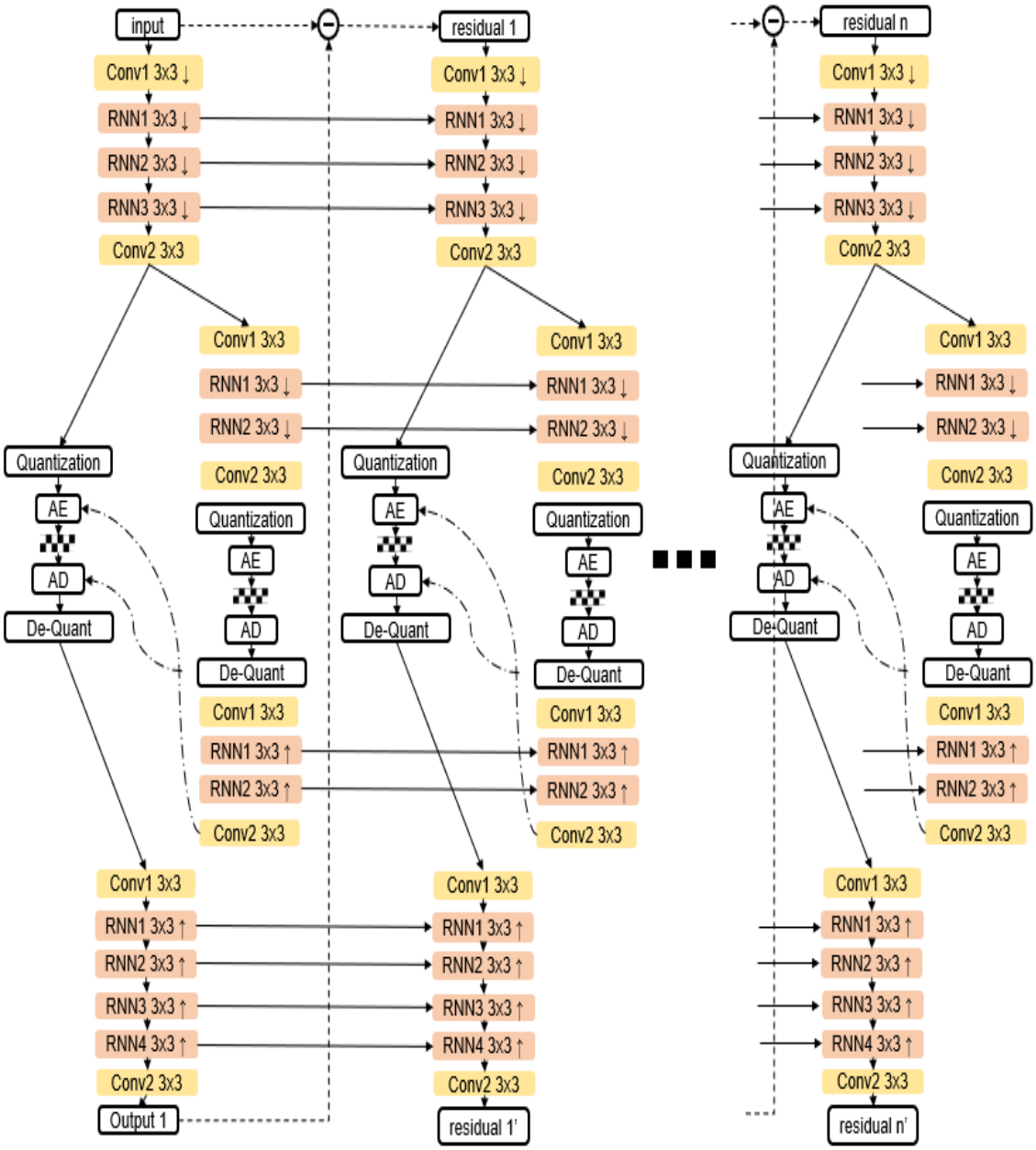


3.2.2 RNN-based architecture with Hyperpriors

For the further optimize the bit rate and avoid reweighting during training. We add an RNN-based hyperprior to further compress the encoded data and generate the output from hyper-decoder. The pass the hyper-decoded data into baseline entropy coder, to optimize the bitrate.

As shown in the Fig.13, the encoded data pass into two ways, one is the quantization and entropy coder layer, and another is the hyperprior network. RNN-based hyperprior architecture is designed like the baseline architecture, consists encoder, decoder,

Fig. 13: The multiple layers of RNN-based image compression architecture with RNN-based hyperpriors



quantization and entropy coder. The hyper-encoder has two CNNs, two RNNs, a quantization and an entropy encoder, the two RNNs are for down sampling. Down sampling the feature map size into $\frac{1}{2}, \frac{1}{4}$. The hyper-decoder has two CNNs, two RNNs,

which is symmetric architecture of hyper-encoder. Besides, there are also two depth-to-spaces behind each RNN for up sampling.

3.2 Quantization

In conventional image compression coding usually uses a round-based function, and the quantized digital value is located at the center of the integer bins, which can be presented by the equation as $\hat{y} = round(y)$, where y is the representations generated by encoder, and the $\hat{y}$ is the quantized data by using round-based quantization. Since the derivative of the round-based quantization function itself is almost get 0 everywhere, it is impossible to perform effective backpropagation directly into the network. In the works[14] added an additive uniform noise instead of quantization. And in the works[21], they directly use the sign function as the binarization,

$$Sign(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{24}$$

For this method also has some limitation, therefore, in our works, during the process of training, adding uniform noise as quantization, that is $\hat{y} = y + \theta$, where θ is random uniform noise. Enhance, we use as $\hat{y} = round(y)$ as the quantization operation in the test stage.

3.3 Rate-distortion Optimization

For lossy compression, we want the bit rate to be as small as possible and the reconstructed image quality to be as high as possible. The traditional autoencoder structure that only constrains the output does not optimize the entropy of the intermediate data. There are two methods, one is to maximize the entropy to make the reconstruction quality as high as possible, and the other is to minimize the entropy while maintaining the reconstruction quality. The two conflict with each other, so it is necessary to weigh the image quality loss caused by the bit rate and compression to find the optimal result under the balance[22]. Therefore, a more complete loss function can be expressed by the following formula,

$$L = \lambda * L_d + L_R \tag{25}$$

where, the L is the loss, L_d is distortion that can set as MSE, L1 loss or even MS-SSIM. The L_R is the actual rate.

Under the structure of autoencoder, what we want to store is the intermediate data generated by the encoder. This data will become the actual bit stream after entropy coding. Therefore, by calculating the entropy of the intermediate data, the concentration degree of the intermediate data can be approximated, and the calculation formula of the entropy

3.3.1 Multiplayer without hyperprior

In order to realize the scalable coding, we allocate the bits to multiple layers, by adjusting the layer-wise lambda values in Lagrangian multiplier-based rate-distortion optimization function.

$$L = \sum_{i=1}^{N} (\lambda_i * L_{di} + L_{Ri}) \tag{26}$$

where, we set different λ values in each layer. As for the experiment, we train four layers, and each layer set λ as [0.001, 0.002, 0.004, 0.008]

3.3.2 Multiplayer with hyperprior

For the RNN-based architecture with hyperprior, not only generate the bitstream from encoder, also generate the bitstream from hyper-encoder.

$$L = \sum_{i=1}^{N} (\lambda_i * L_{di} + (L_{yi} + L_{zi})) \tag{27}$$

where, L_{di} is the distortion of each layer, L_{yi} is the bitrate of encoded, L_{zi} is the bitrate of hyper-encoded.

Chapter 4

Experiments and Results

4.1 Overview

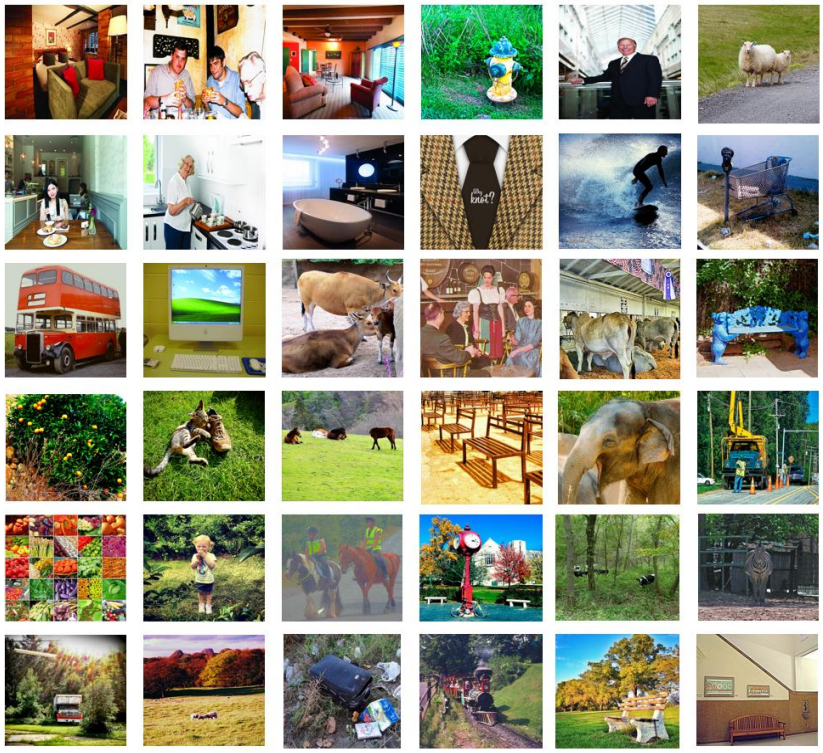
In this chapter, we show the comparison of our work and traditional image compression algorithm and existing deep learning-based image compress method. We show the design of our experiments and visualization of reconstructed images.

4.2 Experiment Setting

4.2.1 Dataset

The dataset includes the kodak dataset[17] containing 24 uncompressed images, and the MS-COCO dataset[18] contains 1.18 million images, the sample of MS-COO dataset shown in the MS-COCO dataset was cropped into 256x256 blocks for training. Kodak's uncompressed 24 images were used for testing.

Fig. 14: MS-COCO dataset samples



4.2.2 Training details

With Adam Optimizer[7], the initial learning rate is set to 0.0001. For RD optimization, set the rate loss to 0 in the initial stage, and after the network has fully converged, gradually increase the weight of the rate loss, so that the distribution of intermediate data is gradually concentrated. This can prevent the network from quickly converging to a local optimum with a low entropy value and cannot fully learn the characteristics of the image distribution. The batch size is 8 and the training step is 400,000.

4.3 Experiment Results

Our work compare with the traditional image compression algorithm such as JPEG, JPEG2000, BPG, and the existing deep learning-based image compression methods, which are Google's work[21] and Balle's work[22]. The results is based on average of Kodak dataset.

4.3.1 RD curves under single layer without hyperprior

First, we present an RNN-based image compression with quantization and entropy coding, different from the Google's work that RNN-based image compression with binarization. To check whether our work can improv the performance or not, we first do the comparison under the baseline (single layer). And the Balle's work also under the baseline. The Fig.15 and Fig.16 show the comparison of PSNR and MS-SSIM, respectively.

The red point is our work which is under the baseline without hyperprior architecture. The 256_512 means the hidden values of RNNs. In the encoder, the first CNN hidden value set as 128, the RNNs hidden values set as [256, 512, 512]. In the decoder, RNNs hidden values set as [512, 512, 256, 128].

The experimental results demonstrate that our work in baseline can be better than JPEG, JPEG2000, and exiting RNN-based image compression method in PSNR. In the MS-SSIM, our work can be better than Balle's work, and approach to the BPG.

Fig. 15: RD curves of PSNR compared baseline of our work with existing image compression methods

Fig. 16: RD curves of PSNR compared baseline of our work with existing image compression methods

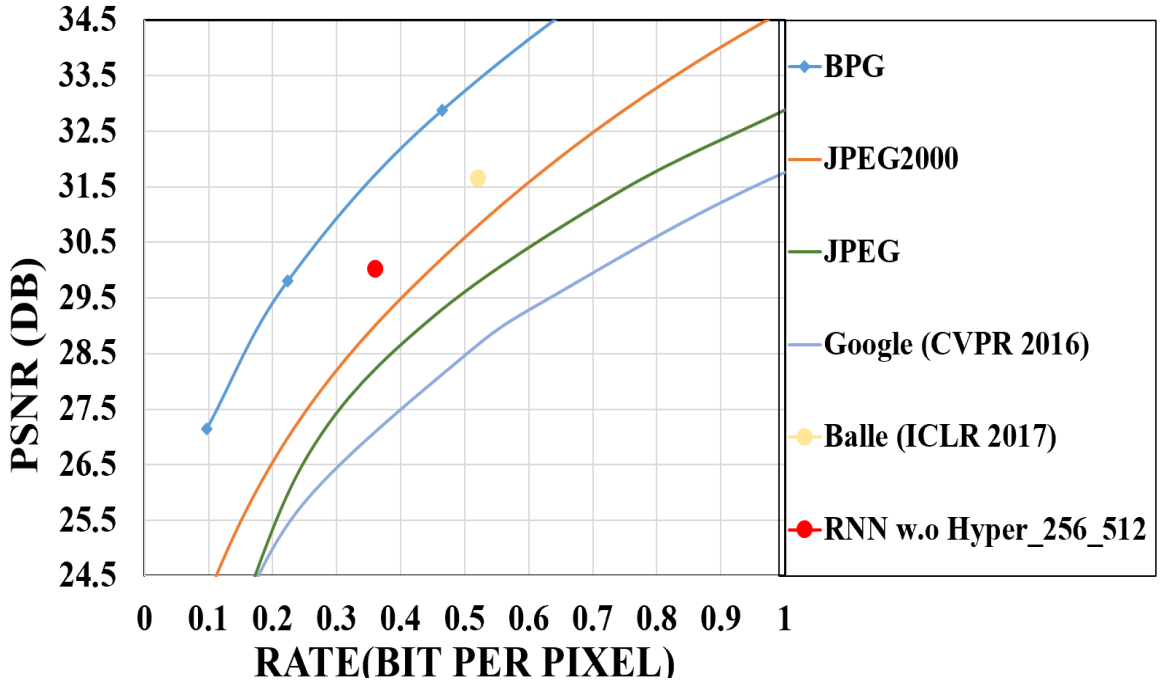
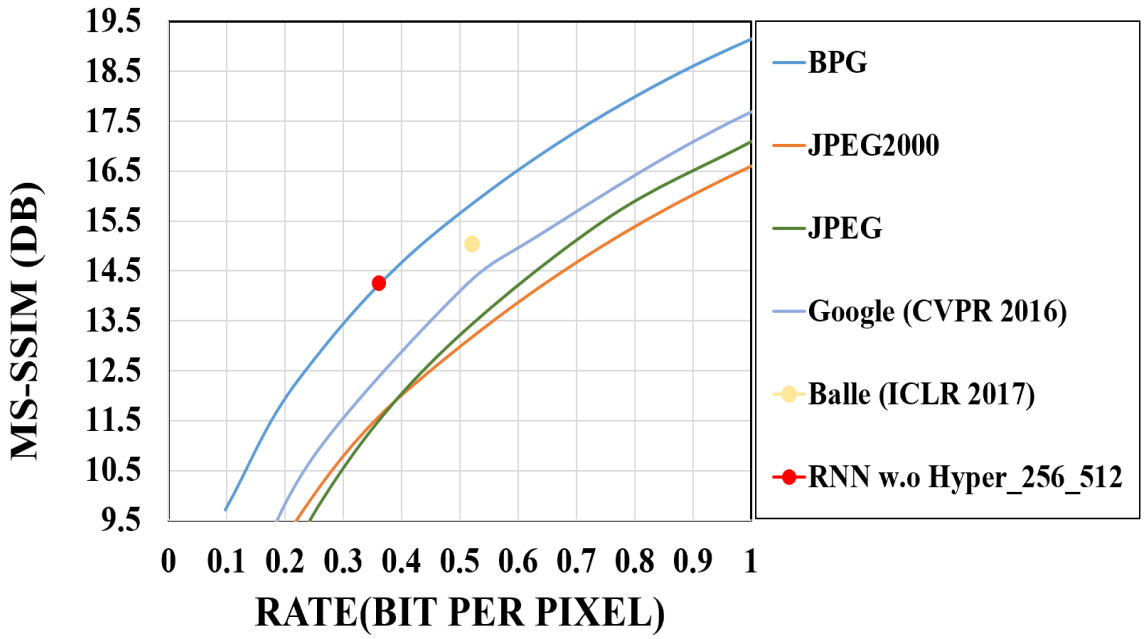


Fig. 17: RD curves of MS-SSIM compared baseline of our work with existing image compression methods



4.3.2 RD curves under multiple layers without hyperprior

For the baseline is performed better than JPEG, JPEG2000, BPG and exiting RNN-based image compression method. So, we try to compare the results under the multiple layers. For our work in multiple layers is scalable coding, that we allocate the bits to multilayer, by setting different lambda values in Lagrangian multilayer-based rate-distortion optimization function. As we test the four layers, and each layer, we set the different λ values. From the first layer to fourth layer, we set λ_1= 0.001, λ_2= 0.002, λ_3= 0.004, λ_4= 0.008, respectively. The results shown in Fig.17 and Fig.18.

The green line is our work. The experimental results demonstrate that our work in four layers can be better than JPEG, JPEG2000, and exiting RNN-based image compression method, even the fourth layer point is a little bit better than Balle's work[23] in PSNR. In the MS-SSIM, our work can be better than Balle's work, and approach to the BPG.

Fig. 18: RD curves of PSNR under 4 layers of our work compare with existing image compression method

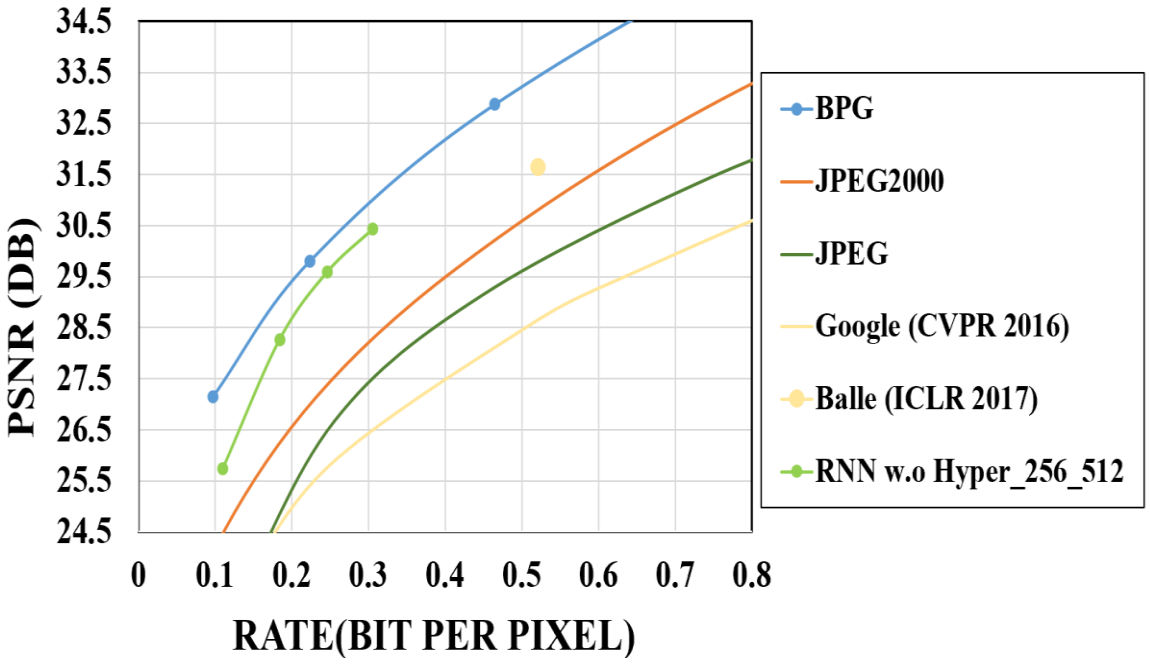
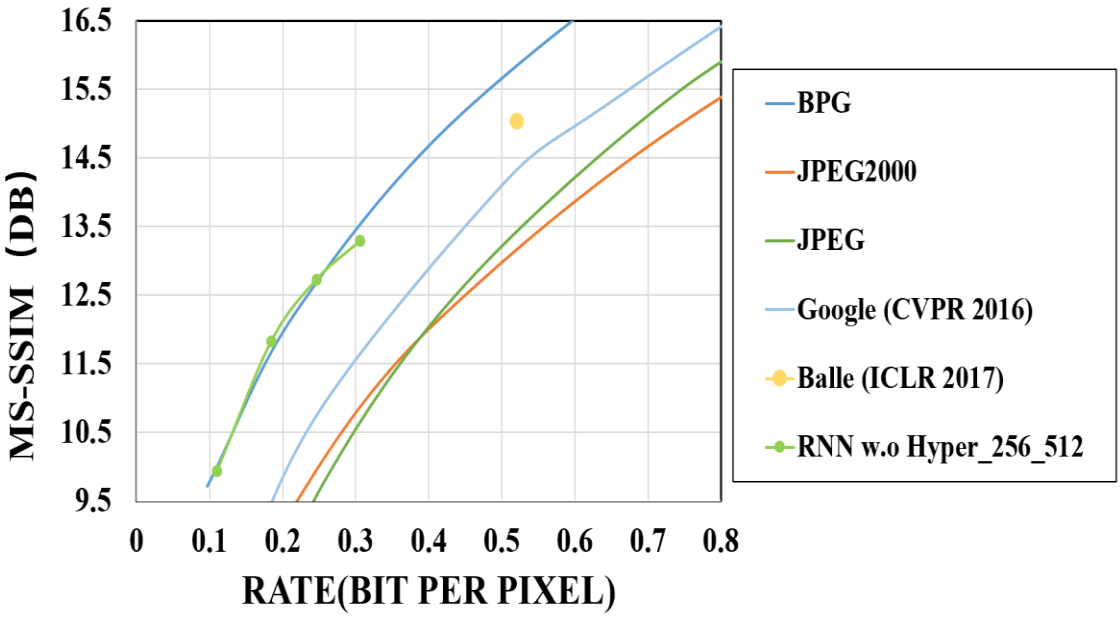


Fig. 19: RD curves of MS-SSIM under 4 layers of our work compare with existing image compression method



4.3.3 RD curves under multiple layers with hyperprior

Because the curve of RNN-based image compression without hyperpriors, has a clear tendency to keep to the right, in order to avoid reweighting and optimize the bitrate at same time, we try to add the hyperprior. For the CNN-based hyperprior cannot fit for scalable coding, so we present an RNN-based hyperprior. The experiment results shown in Fig19 and Fig.20.

The green line is our work without hyperprior, and the red line is our work with hyperprior. The experimental results demonstrate that our work in four layers can be better than JPEG, JPEG2000, and exiting RNN-based image compression method. And add hyperprior can improve the performance, even the fourth layer point is approach to the Balle's work with hyperprior in PSNR. In the MS-SSIM, our work with hyperprior can be better than the BPG.

Fig. 20: RD curves of PSNR comparison of our work with hyperprior and existing image compression methods

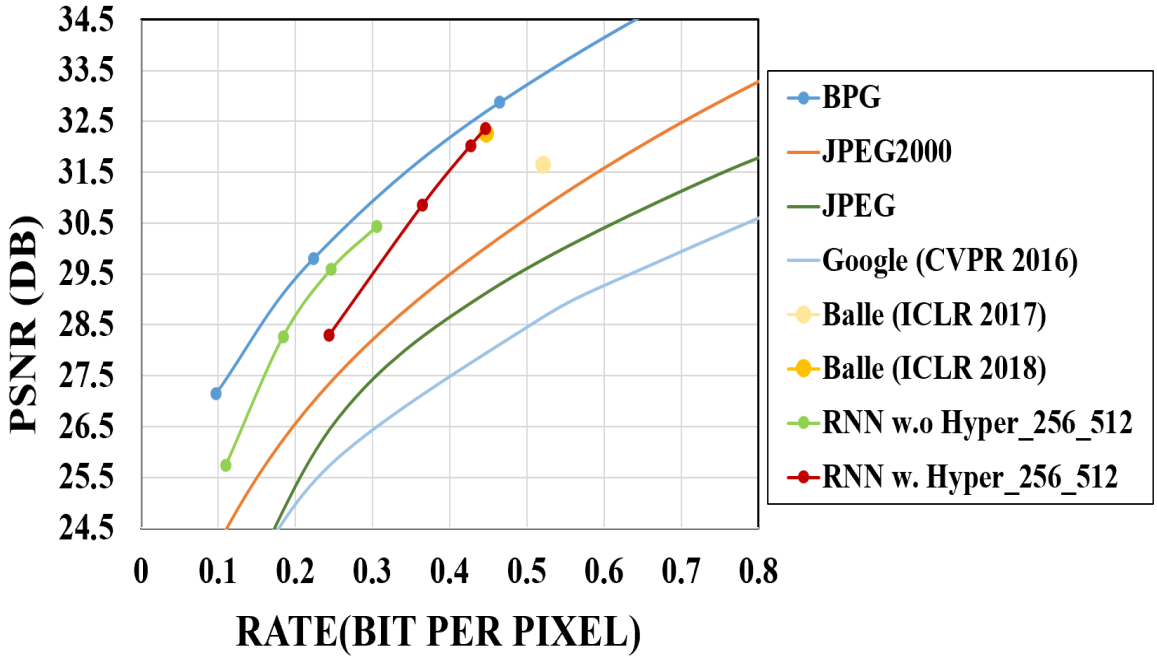
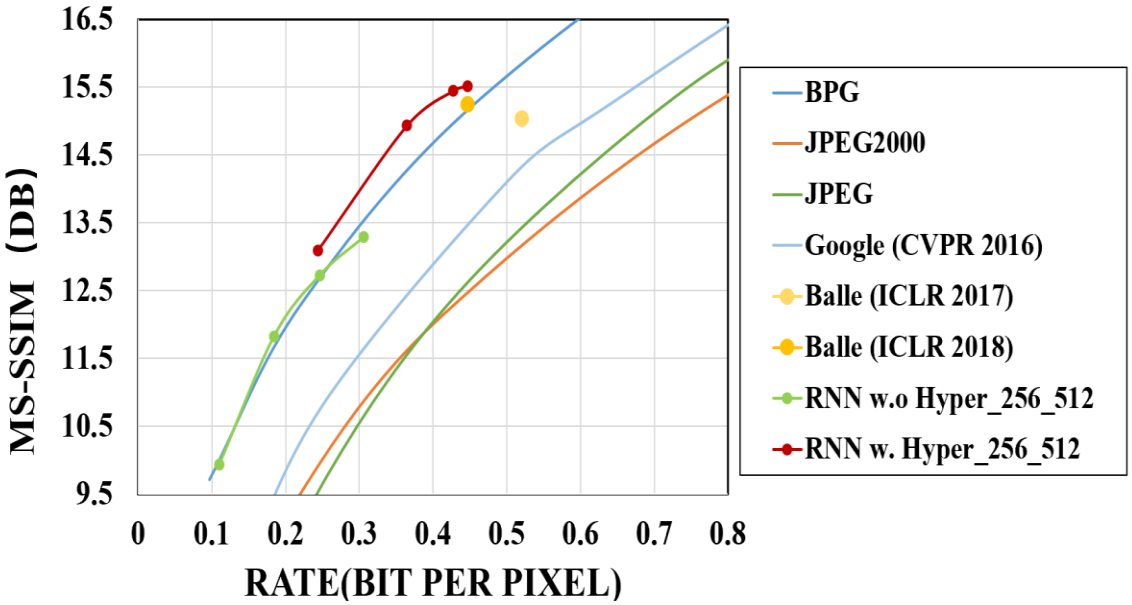


Fig. 21: RD curves of MS-SSIM comparison of our work with hyperprior and existing image compression methods



4.4 Visualization of Reconstructed Images

4.4.1 RNN-based image compression without hyperprior

Fig. 23: Comparison of our work without hyperprior and the existing image compression methods, under Kodak dataset 4th image

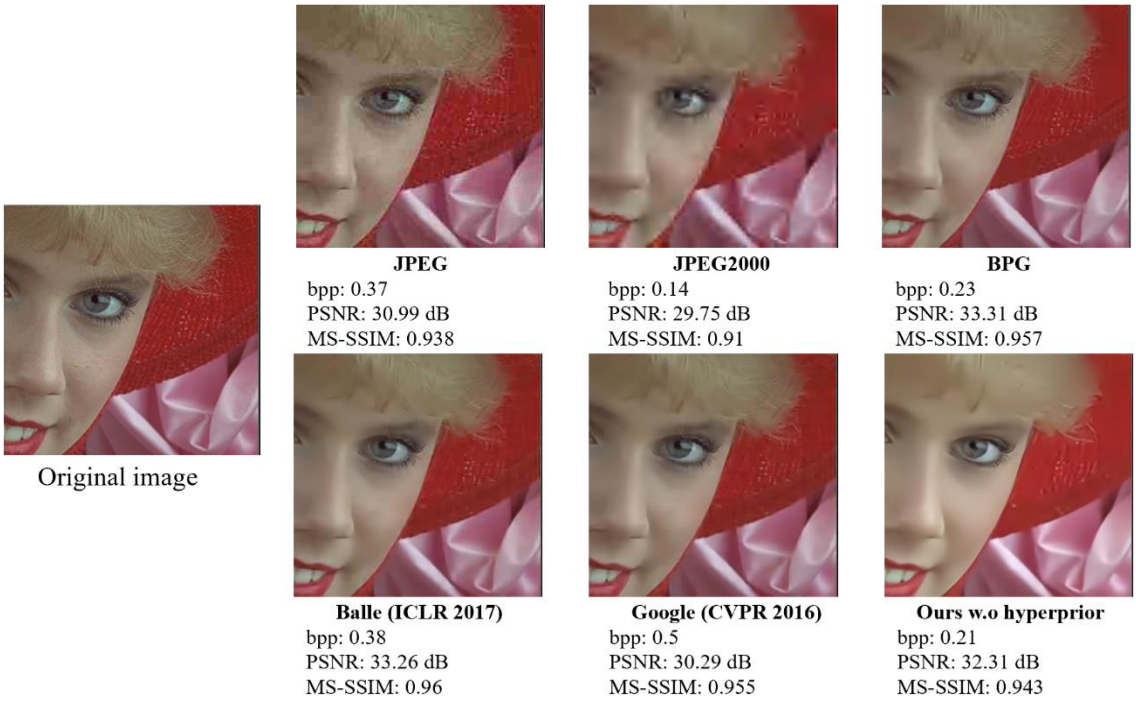


Fig. 22: Comparison of our work without hyperprior and the existing image compression methods, under Kodak dataset 7th image

Fig. 24: Comparison of our work without hyperprior and the existing image compression methods, under Kodak dataset 15th image

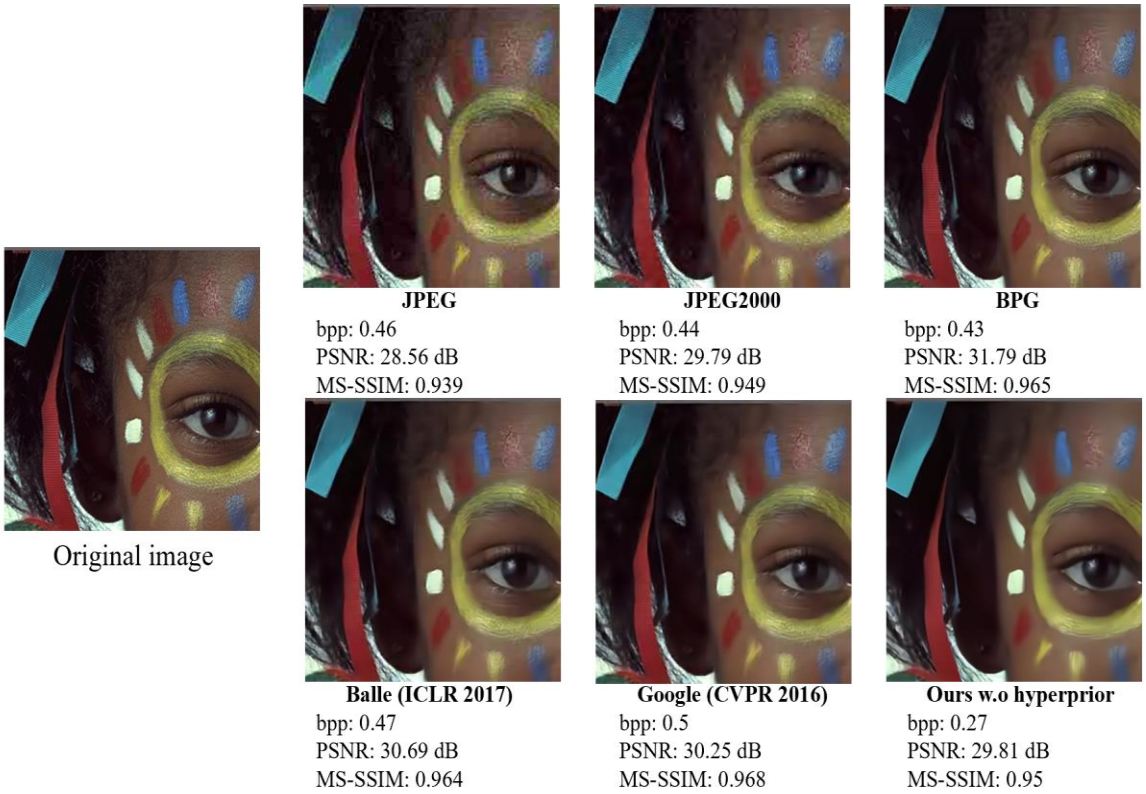


Fig. 25: Comparison of our work without hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.

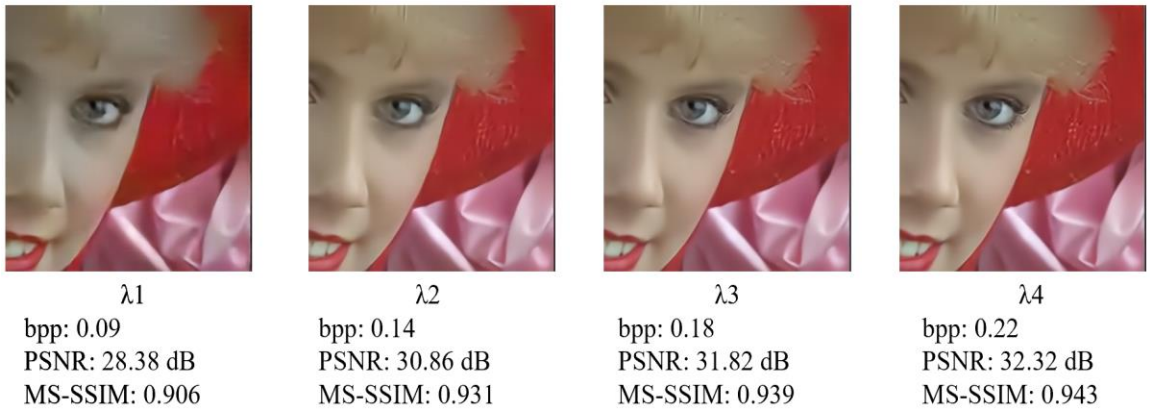
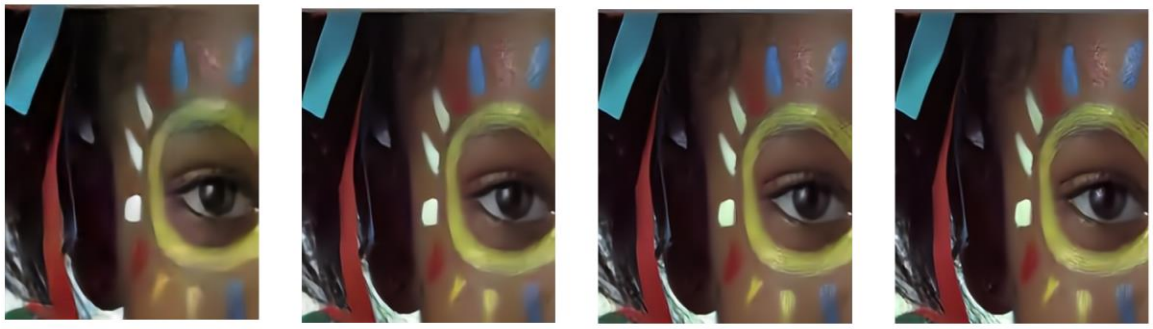


Fig. 26: Comparison of our work without hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.

λ1	λ2	λ3	λ4
bpp: 0.11	bpp: 0.17	bpp: 0.22	bpp: 0.27
PSNR: 26.81 dB	PSNR: 29.43 dB	PSNR: 30.90 dB	PSNR: 31.83 dB
MS-SSIM: 0.94	MS-SSIM: 0.96	MS-SSIM: 0.971	MS-SSIM: 0.975

Fig. 27: Comparison of our work without hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.



λ1	λ2	λ3	λ4
bpp: 0.11	bpp: 0.16	bpp: 0.21	bpp: 0.27
PSNR: 26.84 dB	PSNR: 28.50 dB	PSNR: 29.25 dB	PSNR: 29.82 dB
MS-SSIM: 0.919	MS-SSIM: 0.938	MS-SSIM: 0.945	MS-SSIM: 0.95

4.4.2 RNN-based image compression with hyperprior

Fig. 28: Comparison of our work with hyperprior and the existing image compression methods, under Kodak dataset 4th image.

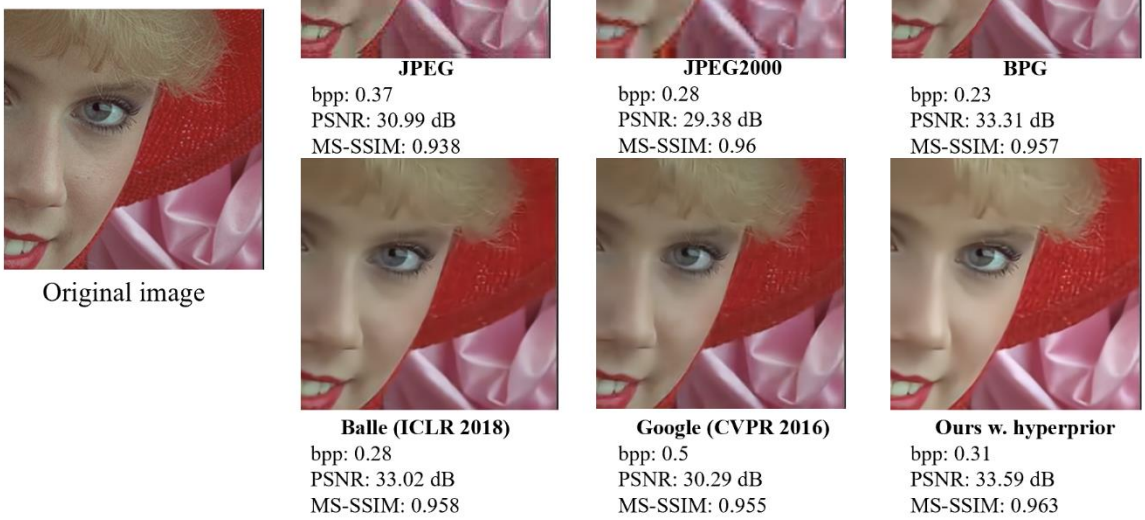


Fig. 29: Comparison of our work with hyperprior and the existing image compression methods, under Kodak dataset 7th image.

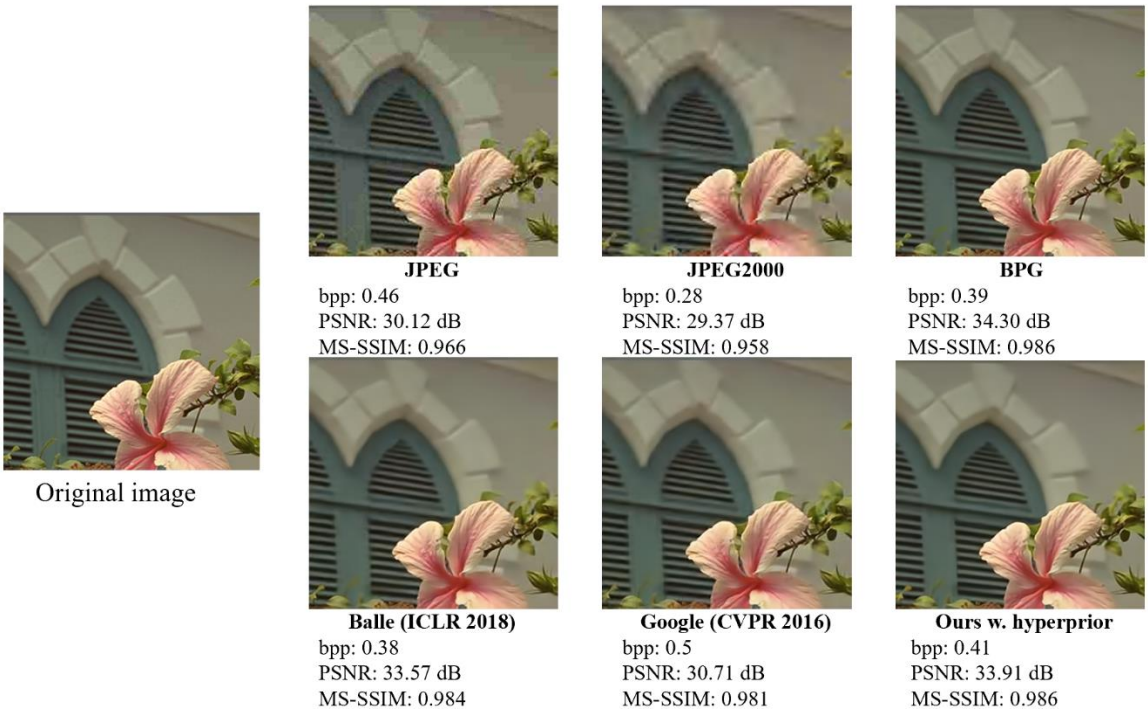


Fig. 30: Comparison of our work with hyperprior and the existing image compression methods, under Kodak dataset 15th image.

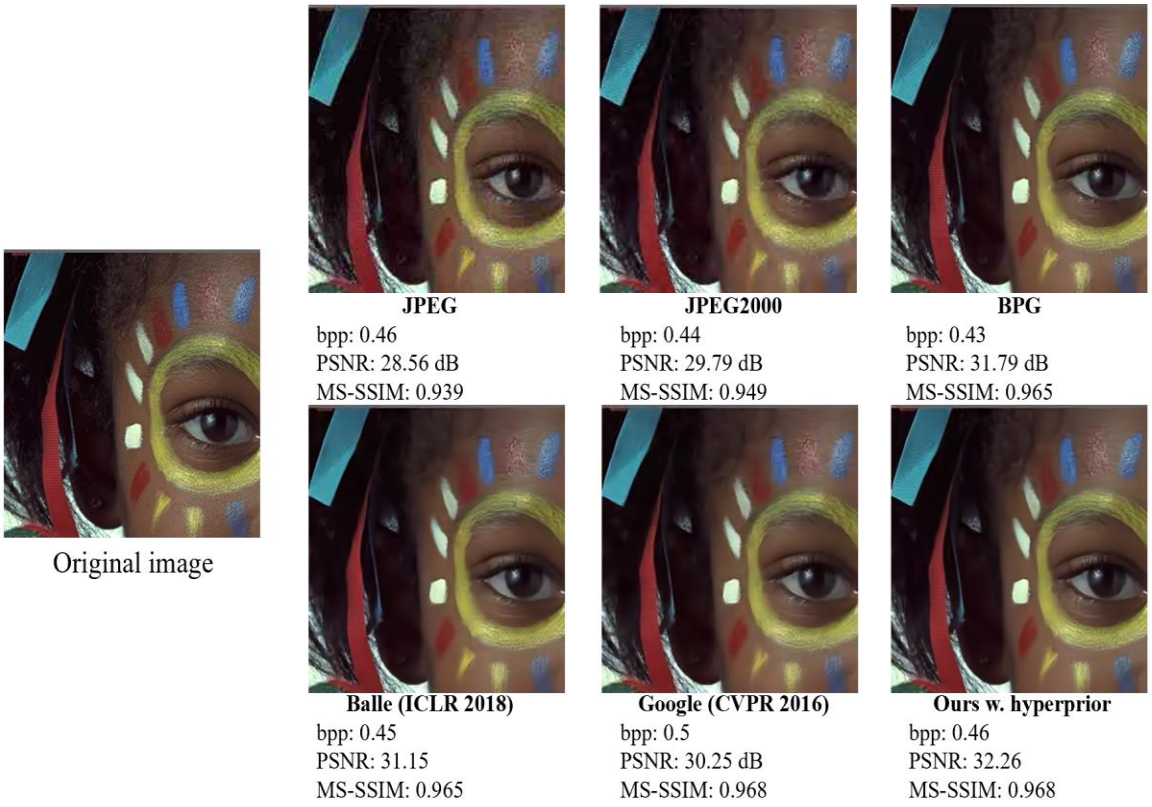


Fig. 31 Comparison of our work with hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.

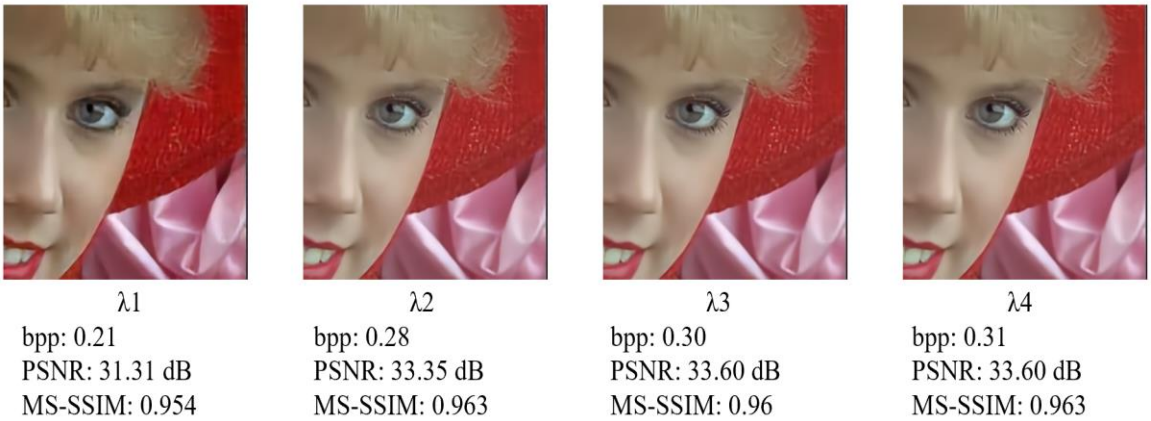


Fig. 32 Comparison of our work with hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.

λ1	λ2	λ3	λ4
bpp: 0.25 PSNR: 29.72 dB MS-SSIM: 0.975	bpp: 0.34 PSNR: 32.44 dB MS-SSIM: 0.983	bpp: 0.39 PSNR: 33.73 dB MS-SSIM: 0.985	bpp: 0.41 PSNR: 33.91 dB MS-SSIM: 0.986

Fig. 33 Comparison of our work with hyperprior of different λ value, each value from left to right is 0.001, 0.002, 0.004, 0.008, respectively.

λ1	λ2	λ3	λ4
bpp: 0.28 PSNR: 28.68 dB MS-SSIM: 0.955	bpp: 0.38 PSNR: 30.29 dB MS-SSIM: 0.965	bpp: 0.44 PSNR: 31.10 dB MS-SSIM: 0.967	bpp: 0.46 PSNR: 31.26 dB MS-SSIM: 0.968

Chapter 5

Discussion

5.1 Discussion of RNN Performance

We do some discussions of RNN architecture. For different RNN hidden value can make different results. We set the encoder RNN hidden value as [64, 128, 128], and [256, 512, 512], the decoder is the symmetric set, respectively. All the work based on the RNN-based image compression without hyperprior. The results shown in Fig.33 and Fig.34. Experiment results demonstrate that the RNN hidden value of [256, 512, 512] can be better than the hidden value of [64, 128, 128] both in PSNR and MS-SSIM. This is why we are using more high RNN hidden value to do all above work.

Fig. 34: RD curves of PSNR of our works compare with the existing image compression method. And our works include two different RNN hidden values-based image compression without hyperprior.

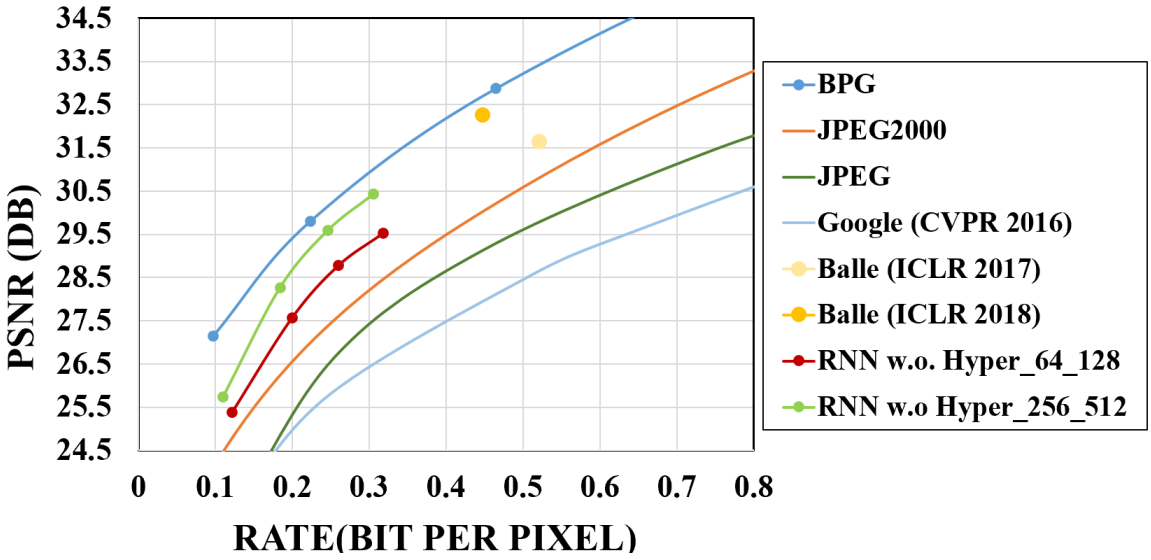
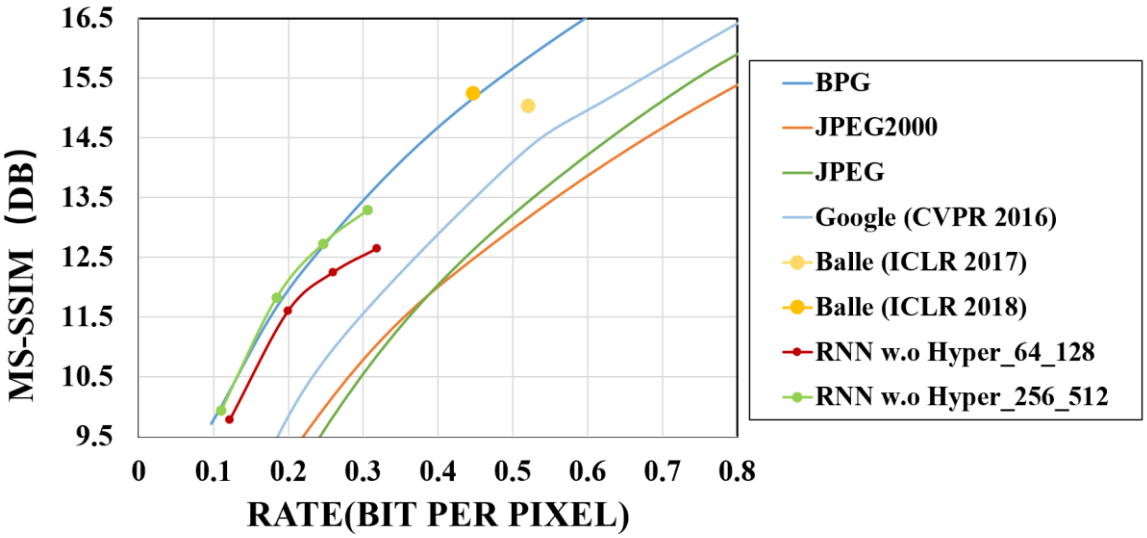


Fig. 35: RD curves of MS-SSIM of our works compare with the existing image compression method. And our works include two different RNN hidden values-based image compression without hyperprior.



5.2 Future Work

Based on this work, two schemes can be tried in the future. First is to change the RNN architecture, for currently we utilize LSTM architecture for image compression. In the future, we can utilize GRU[24] architecture for image compression. As we introduced in chapter 2, the LSTM has three gates to do the computation while GRU is the simplification of LSTM, which only has two gates for computation. Can reduce the amount of calculation and increase the calculation speed and can improve the accuracy based on GRU. Second, we will improve the loss function. We use the rate distortion optimization for optimize loss, and we utilize the MSE for distortion part, in the future, we will try to utilize MS-SSIM as the distortion.

Chapter 6

Conclusion

This paper first introduces the basic concepts of image compression, and then introduces the basic steps and principles of image compression including transformation, quantization, and entropy coding. Then it summarizes a variety of existing traditional image compression methods, including JPEG, JPEG2000, and BPG, and analyzes and introduces the current learning-based image compression methods.

In this paper, first we present an RNN architecture with quantization to make the feature maps of RNNs more compressible. Then, we utilize the entropy coding to further reduce the redundancy and generate bitstream. Then we allocate the bits to multilayer, by setting different lambda values in Lagrangian multilayer-based rate-distortion optimization function. Experimental results demonstrate that better than JPEG2000. Finally, we present an RNN-based hyperprior to optimize bitrate, and change RNN hidden value larger, experimental results demonstrate that adding hyperprior architecture can improve the final performance, and change RNN hidden value larger, the performance can be improved.

Acknowledgement

Firstly, I would like to express my thanks to my thesis advisor, Prof. Jiro Katto. Thanks for his help and encouragement throughout the course of this research. This is the first time I have been involved in this field of research. In this two-year master's career, the professor gave me a lot of meaningful suggestions and improvement programs, which allowed me to progress and grow. I am very fortunate to be able to work and study in the professor's laboratory two years ago. The professor's serious and rigorous attitude has benefited me a lot.

Then, I want to express my special thanks to Zhengxue Cheng, who is a great doctor of our laboratory. Thanks for her academic guidance, I learned a lot of valuable knowledge in my two-year master's life. And thank all the laboratory members for taking care of me, let me have a pleasant study life in Japan.

Finally, I would like to thanks to my families and friends who love me and support me a lot. Without their understanding and support, I would never achieve so far.

Reference

[1] G. K Wallace, “The JPEG still picture compression standard”, IEEE Trans. on Consumer Electronics, vol. 38, no. 1, pp. 43-59, Feb. 1991.

[2] M. Rabbani, R. Joshi, “An overview of the JPEG2000 still image compression standard”, ELSEVIER Signal Processing: Image Communication, vol. 17, no, 1, pp. 3-48, Jan. 2002.

[3] Shaikh, P. Gadekar, “Huffman Coding Technique for Image Compression”, COMPUSOFT, an international journal of advanced computer technology, April-2015

[4] Adjeroh, Donald A., and Moon-Chuen Lee. "Scene-adaptive transform domain video partitioning." IEEE Transactions on Multimedia 6.1 (2004): 58-69.

[5] Witten, Ian H., Radford M. Neal, and John G. Cleary. "Arithmetic coding for data compression." Communications of the ACM 30.6 (1987): 520-540.

[6] Moon, Yong Ho, Gyu Yeong Kim, and Jae Ho Kim. "An efficient decoding of CAVLC in H. 264/AVC video coding standard." IEEE Transactions on Consumer Electronics 51.3 (2005): 933-938.

[7] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[8] Huynh-Thu, Quan, and Mohammed Ghanbari. "Scope of validity of PSNR in image/video quality assessment." Electronics letters 44.13 (2008): 800-801.

[9] Wang, L-T., et al. "SSIM: a software levelized compiled-code simulator." Proceedings of the 24th ACM/IEEE Design Automation Conference. 1987.

[10] Moorthy, Anush Krishna, and Alan Conrad Bovik. "Visual importance pooling for image quality assessment." IEEE journal of selected topics in signal processing 3.2 (2009): 193-201.

[11] Fujita, Mitsutaka, et al. "Peculiar localized state at zigzag graphite edge." Journal of the Physical Society of Japan 65.7 (1996): 1920-1923.

[12] Taubman, David. "High performance scalable image compression with EBCOT." IEEE Transactions on image processing 9.7 (2000): 1158-1170.

[13] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard”, IEEE Transactions on Circuits and Systems for Video Technology, August 2003

[14] L. Theis, W. Shi, A. Cunninghan and F. Huszar, “Lossy Image Compression with Compressive Autoencoders”, Intl. Conf. on Learning Representations (ICLR), pp. 1-19, April 24-26, 2017.

[15] Tatwawadi, Kedar. "Deepzip: Lossless compression using recurrent networks." URL https://web. stanford. edu/class/cs224n/reports/2761006. pdf (2018).

[16] Xingjian, S. H. I., et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." Advances in neural information processing systems. 2015.

[17] Kodak dataset: http://r0k.us/graphics/kodak

[18] MS-COCO dataset: http://cocodataset.org/

[19] G. Toderici, S. M.O’Malley, S. J. Hwang, et al., “Variable rate image compression with recurrent neural networks”, Intl. Conf. on Learning Representations (ICLR), pp. 1-12, 2016.

[20] Johnston, Nick, et al. "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[21] G. Toderici, D. Vincent, N. Johnson, et al., “Full Resolution Image Compression with Recurrent Neural Networks”, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, July 21-26, 2017.

[22] Ballé, Johannes, Valero Laparra, and Eero P. Simoncelli. "End-to-end optimized image compression." arXiv preprint arXiv:1611.01704 (2016).

[23] Ballé, Johannes, et al. "Variational image compression with a scale hyperprior." arXiv preprint arXiv:1802.01436 (2018).

[24] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

[25] L. Zhou, C. Cai, Y. Gao, et al., “Variational Autoencoder for Low Bit-rate Image Compression”, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 2617-2620, June 18-22, 2018.

[26] Choi, Yoojin, Mostafa El-Khamy, and Jungwon Lee. "Variable Rate Deep Image Compression with a Conditional Autoencoder." Proceedings of the IEEE International Conference on Computer Vision. 2019.

[27] W. Shi, J. Cabllero, F. huszar, et al., “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network”, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp.1874-1883, 2016.

List of Publications

[1] Rige Su, Zhengxue Cheng, Jiro Katto, “Quantization Adaptive Recurrent Neural Network for Image Compression”, to be published in the institute of electronics, information and communication engineers (IEICE) March.2019.

[2] Rige Su, Zhengxue Cheng, Jiro Katto, “Optimized Image Compression Based on Recurrent Neural Network”, PCSJ 2019.

[3] Rige Su, Zhengxue, Cheng, Heming Sun, Jiro Katto, “Scalable Learned Image Compression Based on Recurrent Neural Networks”, IEICE general conference 2020.