Learning based image compression (Final Report)

Shiyang Zheng, Yiwen Liang
Jacky Yuan(project mentor)

***Abstract*—Image compression is widely used in today's society. For traditional method like JPEG and JPEG 2000 and previous learning-based compression, most of them can only handle single bit rate or training networks for different compression rates separately. However, in many applications such as data storage, network transmission, they may need a constrain on the bit rate of the image. So, in this situation, a flexible and adaptable image compression method which can deal with variable rate is important. Therefore, in this project, we expect to implement a variable bit-rate deep learning based image compression with an auto-encoder method.**

Keywords—deep learning based image compression, variable bit-rate, auto-encoder

I. OVERVIEW

Image compression is a well-studied problem in the field of image processing. The goal of image compression is to minimize the bitrate based on entropy to represent an image. Nowadays, streaming media industries such as online live broadcasting are becoming more and more popular, which makes the rate control of streaming media is particularly important. As we know, the size of original video material after being shot and edited is too huge to be used directly. In many condition and applications, the transmission network and the storage devices would need the media data to be controlled in a certain bitrate. Moreover, considering the jitter and congestion of the network, the real-time communication applications even need the bitrate during transmission to be dynamic. Based on the cases above, the compression and bitrate control mechanism is highly required. In this report, we are planning to learn and re-implement a fine working deep learning based variable rate image compression method.

According to the project final proposal and our midterm report, from week five to week eight, we basically should do the compression related papers reading and coding knowledge learning. In these week we started our work by learning from the traditional image compression method to find out the general mechanism, then we learnt some well-known implementation of deep learning based image compression (DIC). Finally, we learnt about how to improve the DIC into a variable rate method. Both team members study and reading paper, finding materials, and summarizing separately, then discuss with each other to enhance the understanding of the project. And from week eight to week thirteen, team members should complete the programming implementation of the model which is required for the project and finish the experiment work. In these weeks, Shiyang is basically responsible for Variable rate DIC approach network coding task and Yiwen is doing the test and experiment job. Each team members help each others during the project. And we have achieve our goal that realize a variable rate DIC approach image compression and comparing the variable-rate network with the previous learning based image compression network that can only train and compress images at the same bit rate. And make a discussion and finding result about the performance that we observed. From week fourteen to week fifteen, team members write a project report together according to the completion of the project. And according to the actual presentation requirements and the completion of the project, team members finish their speech drafts and PowerPoint separately, in order to complete the final preparation of the presentation. In the final presentation Shiyang discuss the overview and implementation part and Yiwen talk about the experiment and summery part.

II. PROJECT ACCOMPLISHMENT

In this part, we mainly discuss the knowledge learned through literature reading, and two model that we used in this project, and the experiments we did and the results.

A. Traditional approach

So far there are already many traditional mature image and video compression methods exists. Most of the approaches are transforming coding, such as JPEG and JPEG2000. The traditional way usually achieve the compression goal in 3 steps, I would use JPEG as an example: (Fig. 1)

1. Linearly transform the image, **z = f(x)**. Using DCT (Discrete Cosine Transform) in JPEG would move the DC part to the upper left corner of the matrix and the AC part to the bottom right corner.

2. Using a pre-designed quantization table to quantize the trans-image, **q = Q(z)** . Since human eyes are more sensitive to DC part of the image, the quantization step would make a fine quantization of the DC part and a coarse quantization of DC part. After quantization, the image turns to a series of discrete numbers.

3. Finally, encode the quantized data to bit-stream. For JPEG, DPCM method is used to encode the DC part since the adjacent data usually has slightly difference, Zig-Zag scan and Run-level coding for AC part since there are many 0 values. After that, using Huffman tables to encode both of the two parts. The result is the output compressed image data. (We will ignore this step in the following study because this step generates no loss).

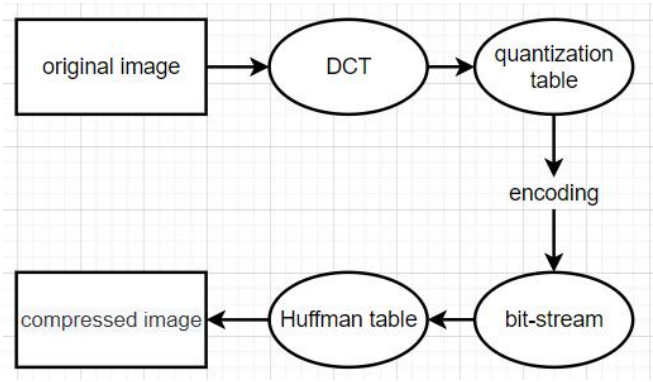

Fig. 1: traditional compression approach

As illustrated above, the traditional way pre-design all the networks and methods applying to all the images, the solution of which is closely related to probabilistic source modeling. However, the step of quantization would introduce error due to the continuous value turned into discrete value. This kind of compression is called lossy compression problem, which can also be solved by deep learning.

From the traditional approach steps, we understood how the image compression works. We started the learning before the course introduces it, to facilitate our following study.

B. Deep Learning Approach

The idea of learning based image compression implementation is roughly the same as that of the traditional method. The difference is that all the transform and quantizer in traditional ways are handcraft designed, but the same part in Learning based method is learnt by machine based on the input training dataset. (shown as figure 2) The transforming and de-transforming functions in encoder and decoder corresponding to the traditional approach like **z = f(x, θ)** and **x' = g(z, φ)**, where z represents for the transformed image, x' represents for the reconstructed image, Θ and φ represent for the two sets of parameters in the CNN layers need to be learnt by deep learning.

The quantization step in deep learning is simplified by rounding, **q = [z]**. Based on [1], to allow optimization via stochastic gradient descent, q is replaced by z with an additive i.i.d uniform noise source **dz ~ U[-1/2, 1/2]**. The quantized data finally is **z' = z + dz**. Noted, we also do not need to de-quantize the data like q in traditional approach when reconstruct the image. In deep learning approach, we directly use z' to generate the bit stream and reconstruct the image. The function **x' = g(z, φ)** turns to **x' = g(z', φ)**.

Based on[1], the learning target is to optimize the trade-off of the entropy of the discretized representation R (rate) and the error arising from quantization D (distortion).

$$\Theta,\varphi = \text{argmin}_{\Theta,\varphi}(R + \lambda D) \quad (1)$$

The rate is obtained by evaluate the entropy of z'. The distortion is assessed between reconstructed image x' and original image x, using mean squared error (MSE). By balancing these two factors using a parameter λ, we can finally get the ideal compression network.

Further study of learning based image compression also adds hyperprior to eliminate the consistency in spatial domain [2] to train the data better.

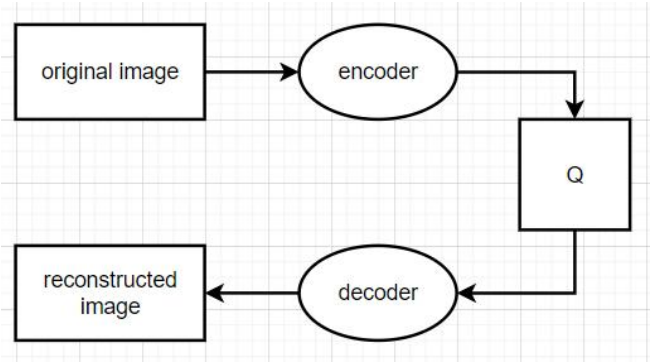

Fig. 2: deep learning approach

We have learnt a lot by learning DIC, both of us have not learn much about deep learning before, learning these knowledge and implement some of it costs a lot of our time but worth it. After learning about this part, we can understand the following part much easier.

C. Variable rate DIC approach

As we learnt above, the DIC optimizes the model by the R-D trade-off, which means achieving different rate means that we need to train several individual model to implement each one of them, which needs a large memory to store all the models. Considering this situation, a variable rate DIC method is necessary.

An approach in [3] solve this problem by scaling the transformed image z using desired bitrate before quantization. (Fig.3) However this method did not consider the quality of the final representation of the image. Because the one model it used is trained based on one R-D trade-off, when we change the bitrate directly, the distortion it represented is not the same as the model trained by the target R-D trade-off.

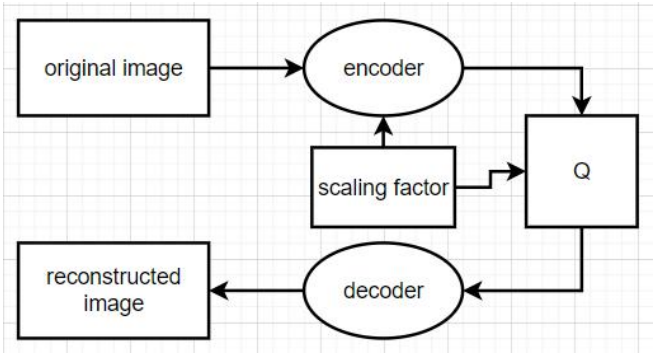

Fig. 3: auto-encoder with scaling factor

Inspired by the deep learning steps, if we need to train one model that satisfied multiple bitrates, we could also make the bitrates as an input dataset. The network takes an image dataset and variable bitrate (we can replace bitrate by parameter λ using in R-D trade-off), then generate one model that takes an image and a parameter in advance to output compressed image, the trade-off function above would turns to:

$$\Theta,\varphi = \sum\lambda\ w^{*}\text{argmin}_{\Theta,\varphi}(R + \lambda D) \quad (2)$$

Where w represent for the weight of each bitrate of the whole consideration. We can also simplify this function by set w to 1 to make all the bitrate equally important.

Article [4] gives a great idea of model using the trade-off above to optimize. They implement the problem by using a modulated autoencoder (MAE). Combining an modulated network implemented by two full connected layers with each CNN layers implemented by [1] mentioned above. The modulated network takes a scalar λ as input and output a sets of modulating and demodulating functions {mi(λ)} and {di(λ)}, then added them into each CNN encoder layers and decoder layers {zi(λ)} and {ui(λ)}, to modulate the network by the parameter λ. (Fig.4)

The parameters need to be learnt in the modulating network are α and β, so the modulating and demodulating functions {mi(λ)} and {di(λ)} turn to {mi(λ, α)} and {di(λ, β)}. The final trade off becomes:

$$\Theta,\varphi,\alpha,\beta = \sum\lambda\ w^{*}\text{argmin}_{\Theta,\varphi,\alpha,\beta}(R + \lambda D) \quad (3)$$

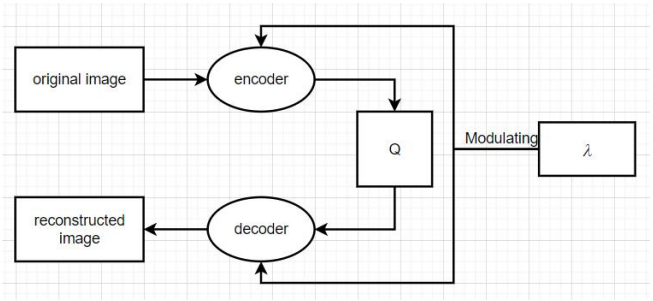

Fig. 4: network with modulated auto-encoder

We plan to implement a variable rate DIC model based on [4] and improve it if possible. Also, we would like to compare the quality of this implementation with the traditional way and DIC way model mentioned above.

D. Network Model implementation

Our implementation is written in python. In this project, we implement two models. The first model is the re-implement of the DIC architecture proposed in [1]. (Fig.5) The purpose of this implement is to compare the performance with the variable rate DIC as the contrast experiment.

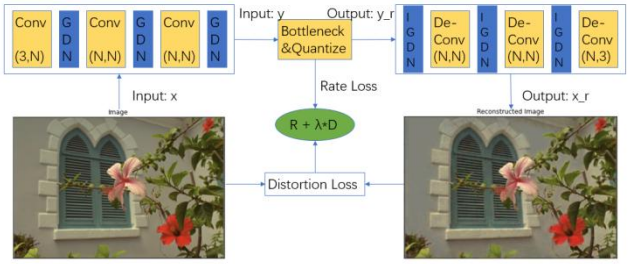

Fig. 5: single DIC approach network architecture

In Fig.5, the encoder network consists of 3 stages of convolution layer and GDN layer. The original network implemented in[1] has a down sample layer in each stage. However, we train the model with the image size 256*256, so there is no need to down sample the data anymore. The decoder network consists of 3 stages of corresponding de-convolution layer and IGDN layer to reconstruct the decoded data back to image.

The GDN is generalized divisive normalization, it is used to normalize the data. The concept is proposed from [1] which is designed to replace the batch normalization layer. The original image input has 3 channel corresponding to R, G and B. The first convolution layer in the encoder expand it to N channels and process the N channels in the following layers. The output from the encoder enter a bottleneck for auxiliary to further lower the consistency between the data for better performance. To reconstruct the image, the last de-convolution layer would turn N channels back to 3 channels.

The loss function for this architecture is R-D loss, we use a parameter λ to govern the trade-off between rate loss and distortion loss. The rate loss is the entropy loss calculated from the data likelihood outputs after quantize. The distortion loss is assessed by mean squared error between the original image and the reconstructed image. Before we train the model, we need to define the loss function using λ. When λ is larger, the weight of distortion loss is higher, which means the optimizer would focus on the distortion more than rate. The reconstructed image would has a better quality but a higher bitrate. Similarity, when λ is smaller, the learning process would concentrate on a lower bitrate. Therefore, we can train several different rate model by adjusting the λ in the loss function.

Intending to design a variable rate DIC, our second implementation refer to the architecture in [4]. The basic architecture is the same as the first model introduced above, however, we add a modulate layer to modulate the convolution layer output of each stages.(Fig. 6)

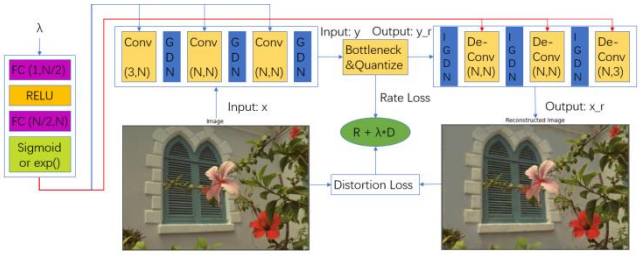

Fig. 6: variable rate DIC approach network architecture

The modulate layer takes a scalar parameter as input and output an N length vector, N corresponding to the amount of channels. The input scalar first go through a fully connected layer and expand to a N/2 length vector, then it go through a relu layer to restrict the parameters non-negative. After set all parameters non-negative, the vector go through a fully connected layer again to expand the length to N. Finally exp function or sigmoid layer is used to normalize the data into a range. By doing so, we can achieve a variable rate image compression.

E. Experiments and result

We evaluate our model by using coco dataset as the training dataset which contains 512 training images, and we test our models on Kodak dataset with 24 test images.

Our first model is implemented by setting $\lambda \in [64, 256, 1024, 4096]$, respectively. Because we need to train a separate network to achieve image compression for different λ value, we train four different networks totally. For each network, we set channel size 128 and batch size 1, and we train 20 epoch. Because the lambda we used in loss function is pre-set and unchanging, in this model we need to train 512 images for each epoch.

Our second model is implemented by setting $\lambda\_set = [64, 256, 1024, 4096]$, as a set. In order to achieve variable rate, we need using these λ as an input when calculating the RD_loss, so for each epoch each image needs to train four times, and totally we need to train 2048 images each epoch. Besides, when we input λ into modulate layer, we need to normalize them, so we use normalize trade-off as $\lambda\_input = \lambda / \max(\lambda\_set)$ and the demodulated part follows the same architecture. Also, for the second model, we set channel size 128 and batch size 1, and by training the network 80 epoch we get a relatively ideal result. Also we have tested a different situation when batch size is equal to four, however, due to the fact that it did not have a better result than batch-size 1, we are not discuss batch-size 4 situation here.

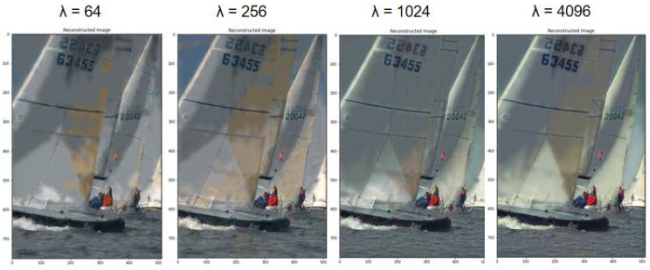

Fig. 7: reconstructed images in different λ value for first model

TABLE I. EVALUATION OF FIRST MODEL UNDER DIFFERENT λ

	PSNR	MS-SSIM	BPP
λ = 64	24.5593	0.8026	0.1317
λ = 256	25.8503	0.8294	0.3007
λ = 1024	23.4678	0.8415	0.7090
λ = 4096	26.1979	0.8522	1.7309

We use MS-SSIM and PSNR to evaluate the image distortion, and use BPP to evaluate the compression image's bit-rate. Fig. 7 shows the reconstructed images in different λ value for first model. In this picture, at any compression bit rate (λ), we all can maintain a good image feature, like the number on the sailboat. And as the increasing in lambda value, the reconstructed image's color becomes brighter and background becomes much clearer, which means the compression image's quality is also improved. Table 1 shows the result numerically. As lambda changes from 64 to 4096, BPP gradually rises from 0.13 to 1.73, and PSNR and MS-SSIM are also gradually increase (except the PSNR value in λ =1024), which means We have to sacrifice some image quality to achieve low bit-rate image compression.

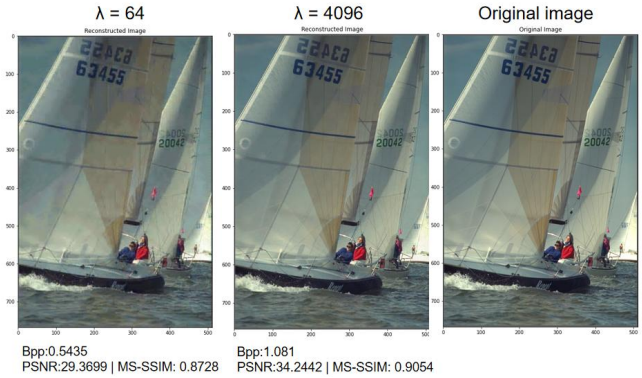

Fig. 8: reconstructed images for low (λ = 64) and high (λ = 4096) bitrates and original image for comparison

When we training second model, Exp function and Sigmoid layer are used separately for testing in the last layer of modulated layer. Because these two results are very similar, Fig. 8 shows the reconstructed images in two different λ value for second model with exp function used in modulated layer. When lambda is equal to 64 and 4096, there is a difference about 0.5 in bpp between them. Therefore, we basically achieve the variable rate image compression. And we have a relatively higher PSNR and MS-SSIM, which means Image features remain good.

III. SUMMARY

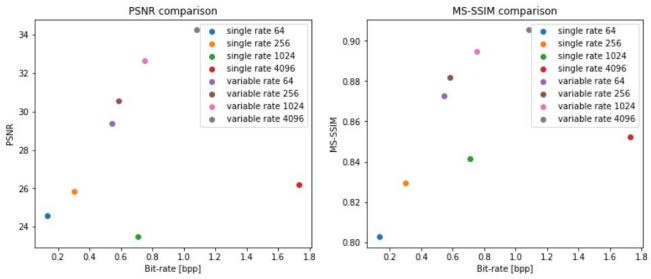

Fig. 9: PSNR and MS-SSIM comparison

From Fig. 9, basically by using a modulated auto-encoder based variable rate deep image compression network, we can achieve a goal that by training one network, a variable rate image compression can be achieved. There is about 0.5 variable rate between the lowest bit-rate and the highest bit-rate in model two. And comparing with single rate network, variable rate network can achieve a better image quality after compression because it's higher PSNR and MS-SSIM. Besides, when we test the variable rate compression network, we found that after training 40 epoch, BPP decreased by about 0.04 per 20 Epochs. So we assume that we can continuing drop the bit-rate by training more times, however, due to time limitation, we did not continue to train more times to test our hypothesis.

By doing this project, we have learnt a lot, such as how to implement a project from finding a topic to searching material and accumulate relevant knowledge, using pytorch to build a deep learning network and by constantly testing and modifying to get a relatively ideal result. And in the future, we should modify the parameters and training more times and adding more lambda value into lambda set, in order to achieve a lower bit-rate which the range is between 0.1 and 0.6 variable rate image compression.

IV. DEPOSITORY OF OUR PROJECT OUTCOME

All our work is in follow link: https://drive.google.com/drive/folders/13fAJK1L_avZ2Y_rfGpBQRYpwpp6uJTzH?usp=sharing

The dataset we used are in "test_dataset" and "train_dataset" folder. The code and result for model one are in "single lambda" folder.

And file "variable_rate_batch1_exp" is the code and result for the second model that we showed the result in our final report, and the model epoch that we trained is in "model_variablerate_batch1" folder. And all other test experiment files and in "others" folder.

REFERENCES

[1] Johannes Ballé, Valero Laparra, Eero P. Simoncelli, " End-to-end Optimized Image Compression," arXiv:1611.01704v3, 2017

[2] Johannes Ballé, David Minnen, Saurabh Singh, "VARIATIONAL IMAGE COMPRESSION WITH A SCALE HYPERPRIOR,"arXiv:1802.01436v2, 2018

[3] L. Theis, W. Shi, A. Cunningham, and F. Husz´ar, " Lossy image compression with compressive autoencoders, " arXiv preprint arXiv:1703.00395, 2017

[4] Fei Yang, Luis Herranz, Joost van de Weijer, " Variable Rate Deep Image Compression with Modulated Autoencoder, " IEEE Signal Processing Letters, 2020, pages: 331 - 335