CONE 240 Machine Learning

Term Project Report

Author: Yichen Pan, Bosen Yang*
UID: 1588445, 1589880

Instructor: Dr. Ying Liu
Santa Clara University
Winter 2020

* Both authors are equally contributed to this report

Task #1

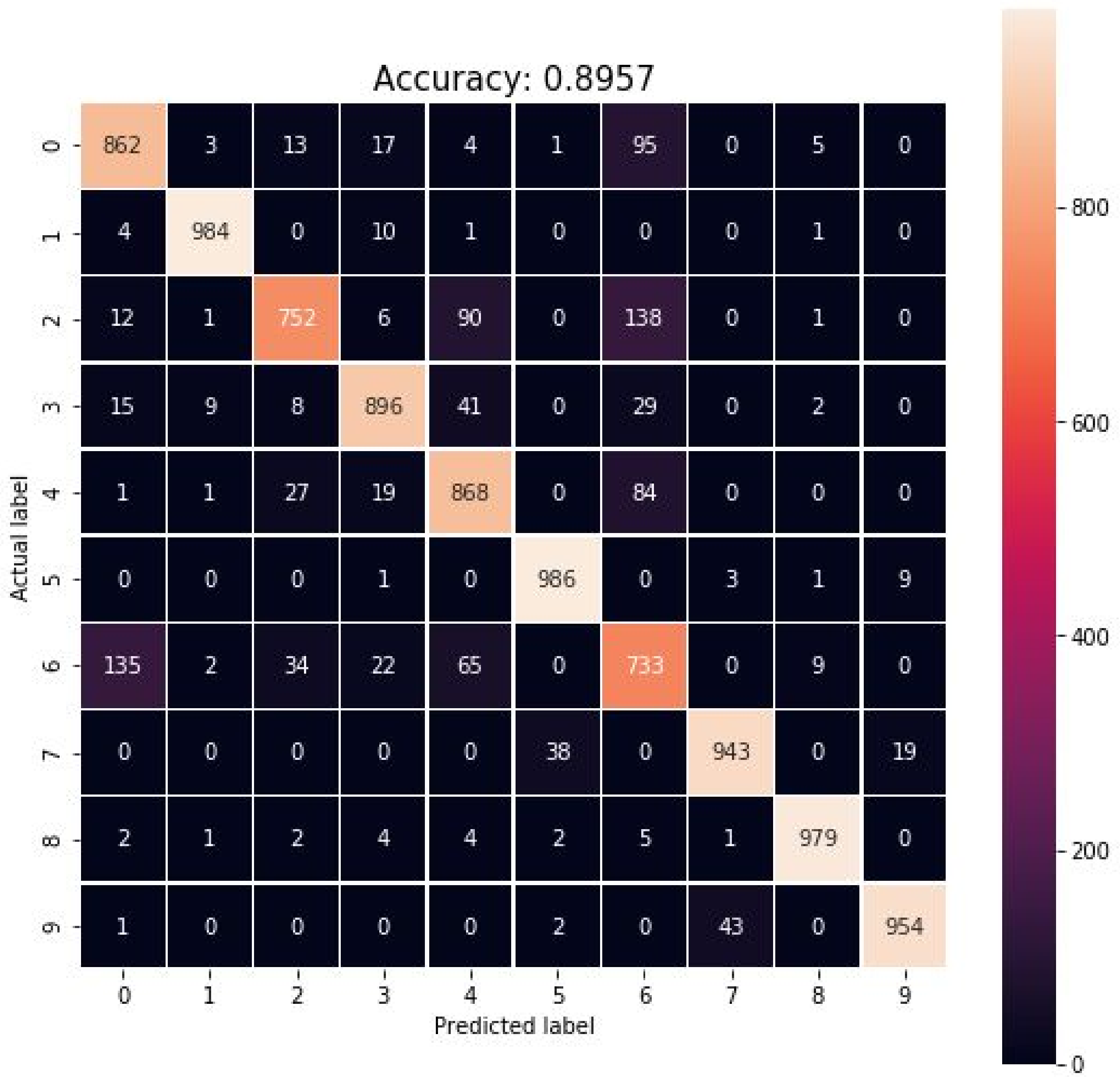

Figure 1.1 Confusion matrix and prediction accuracy of the testing result

The figure 1.1 shows the confusion matrix of the prediction result and overall accuracy of our CNN model on the fashion MNIST dataset. 60,000 samples are used for training and 10,000 samples are used for testing.

Task #2

P	10	50	200
LOSS	0.0138	0.0075	0.0036
PSNR	19.4681	22.3885	25.7363

Figure 2.1 Test Loss and test PSNR when P = 10, 50, 200

2.a

From figure 2.1, a positive correlation can be observed between P, the number of nodes in the compressed layer, and the PSNR. The observation is intuitive since more nodes in the compressed layer means more information can be preserved from the original image during the compression process, which also indicates better accuracy of the decompressed output.

Despite the positive correlation between P and PSNR, a larger P value will also result in the larger size of the compressed images. Considering the goal of image compression is to reduce the data load for transmission, we should prevent sizes of compressed images from getting too large. Therefore, we could argue that higher decompression accuracy comes at the cost of higher transmission data load.

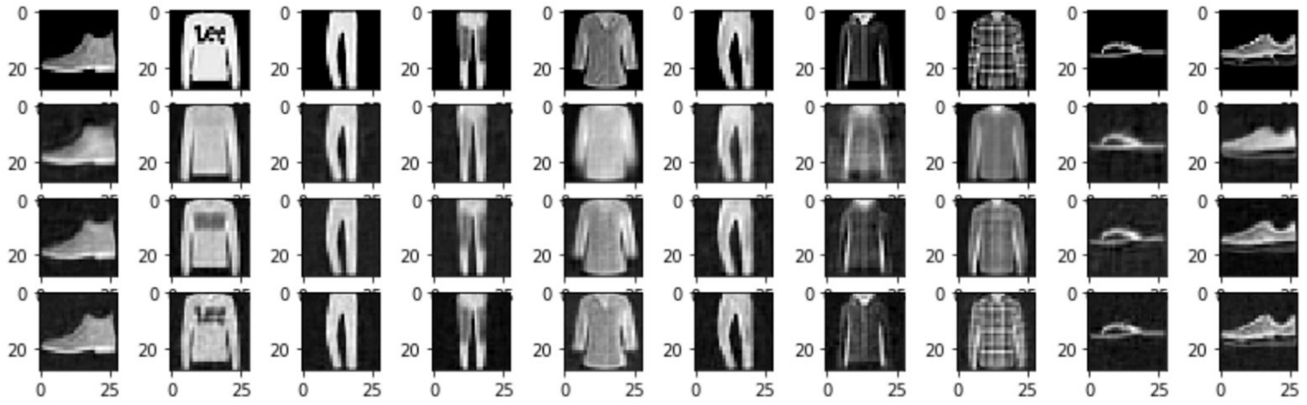

Figure 2.2 Original images (1st row) and compressed images when P=10, 50, 200 (2nd to 4th row)

2.b

From figure 2.2 we can observe that the visual quality of compressed images increases as the P value becomes larger.

With the same P value, images with more detailed textures are more difficult to be compressed. For example, when P10, the letters on the t-shirt on the second column is almost invisible, while the pants on the third column still look almost the same to the original image. Similarly, the check pattern of the shirt on the seventh column was not clearly visible when P = 10 or 50, but the sandals next to it has acceptable visual quality even when P = 10.

The reason behind is the loss of detail information during the compression. In the compression layer, multiple pixels from the original image are connected to one node. Apparently, the information one node can represent is limited. Thus, if we have less nodes in the compression layer (smaller P value), more pixels will be connected to one single node so that more information from the original image will get lost during the compression. Although the decompression layer can upsample the compressed images, it is not able to bring the lost information back.

Task #3

- **Introduction**

In this task, we propose two CNN-based structures to build a color video compression system. Both structures are trained and tested with samples frames from three provided datasets: Basketball Drill, Blowing Bubbles, and Race Horse. The objective is to achieve better reconstruction quality at the same compression ratio, which can be reflected as either higher PSNR value or better visual quality. Also, experiments are performed to study how different compression ratio will affect the reconstruction quality for each CNN. Finally, we discuss possible directions to improve our work in the future.

- **Proposed methods**

In this section we will introduce the architectures of two proposed CNN-based systems. Both systems consist of an encoder and a decoder and were trained in an end-to-end manner. In order to better reflect how different encoder network structure affects the performance, the same decoder structure is used in both systems.

The encoder of our first system is similar to the AlexNet and the encoder of the second system is designed as a simplified GoogLeNet model.

We will first explain the encoder structures of both systems, and then the decoder structure, and finally go into details about how we train the two systems and how to define hyper parameters.

A. Input

The inputs of both systems are blocks with the dimension of 32×24 pixels. By doing this, we will have more flexibility processing frames of different sizes. Apart from that less information loss and faster training speed can also be achieved. For example, if we do a frame wise compression when the frame size is bigger than the input size, we have to clip the frame or tailer the frame in other ways, leading to information loss. As for training speed improvement, we can reduce the computational complexity by reducing the number of convolution operations. For example, if we have a 6×6 frame and do 3×3 convolution with stride 1, we will conduct 16 convolution operations. If we cut the frame by the middle, the number is reduced to 8.

B. AlexNet-like encoder

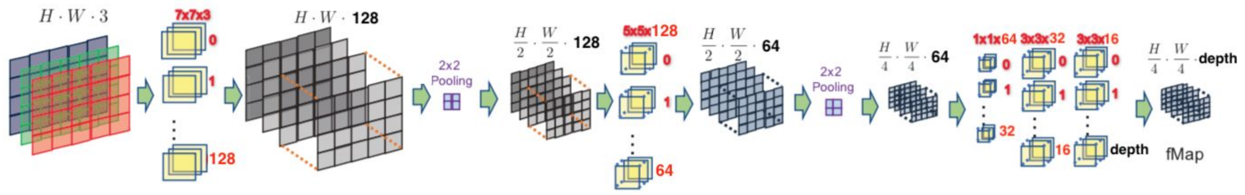
$H \cdot W \cdot 3$
7x7x3
0
1
128
$H \cdot W \cdot 128$
2x2 Pooling
$\frac{H}{2} \cdot \frac{W}{2} \cdot 128$
5x5x128
0
1
64
$\frac{H}{2} \cdot \frac{W}{2} \cdot 64$
2x2 Pooling
$\frac{H}{4} \cdot \frac{W}{4} \cdot 64$
1x1x64
0
1
32
3x3x32
0
1
16
3x3x16
0
1
depth
$\frac{H}{4} \cdot \frac{W}{4} \cdot depth$
fMap

Figure 3.1 Network architecture (i.e., layers, convolutional kernel parameters) of the AlexNet-like encoder

This encoder network structure is derived by stacking multiple convolution layers together, whose depth comes in a decreasing order. Such design serves our purpose of feature selection. We also put max pooling layers in between for dimension reduction.

Type	Filter size / stride	Depth	Output size	Padding	Activation
input			H×W×3		
convolution	7×7 / 1	128	H×W×128	same	relu
max pooling			H/2×W/2×128	same	
convolution	5×5 / 1	64	H/2×W/2×64	same	relu
max pooling			H/4×W/4×64	same	
convolution	1×1 / 1	32	H/4×W/4×32	same	relu
convolution	3×3 / 1	16	H/4×W/4×16	same	relu
convolution	3×3 / 1	compressed layer depth	H/4×W/4×compressed layer depth	same	relu

Table 3.1 detailed information of the AlexNet-like encoder

C. The simplified GoogLeNet encoder

The idea of inception module was applied in this architecture. An inception module is a local network topology where we use parallel filters for pooling and convolution of different receptive fields. The output of the inception module is the concatenation of parallel filter outputs. Additional 1×1 convolution layers are added along with the filters for depth reduction to prevent the output from getting too deep and increasing the computational complexity.

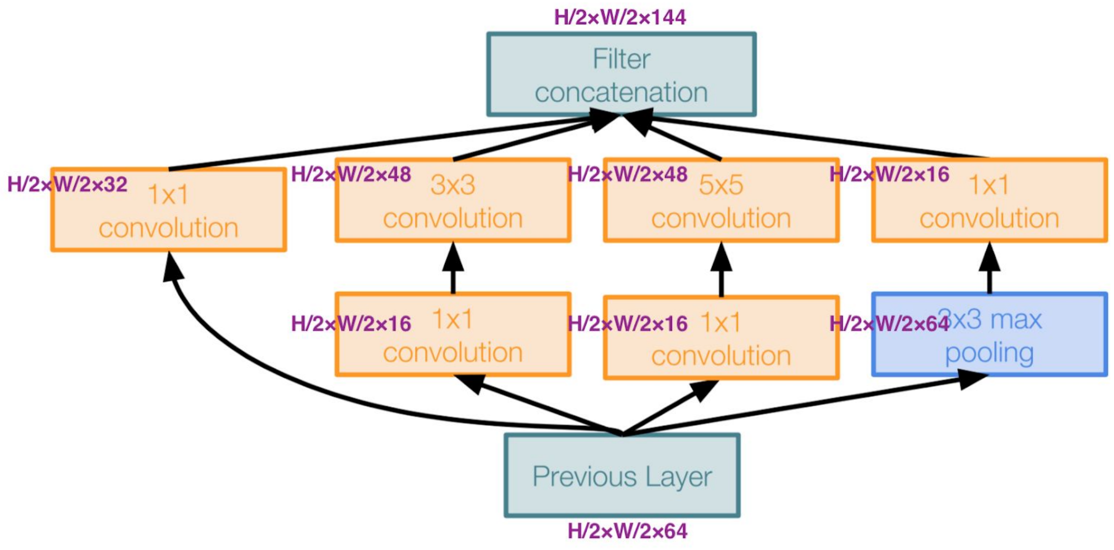

Figure 3.2 1st inception module structure, with dimension reduction

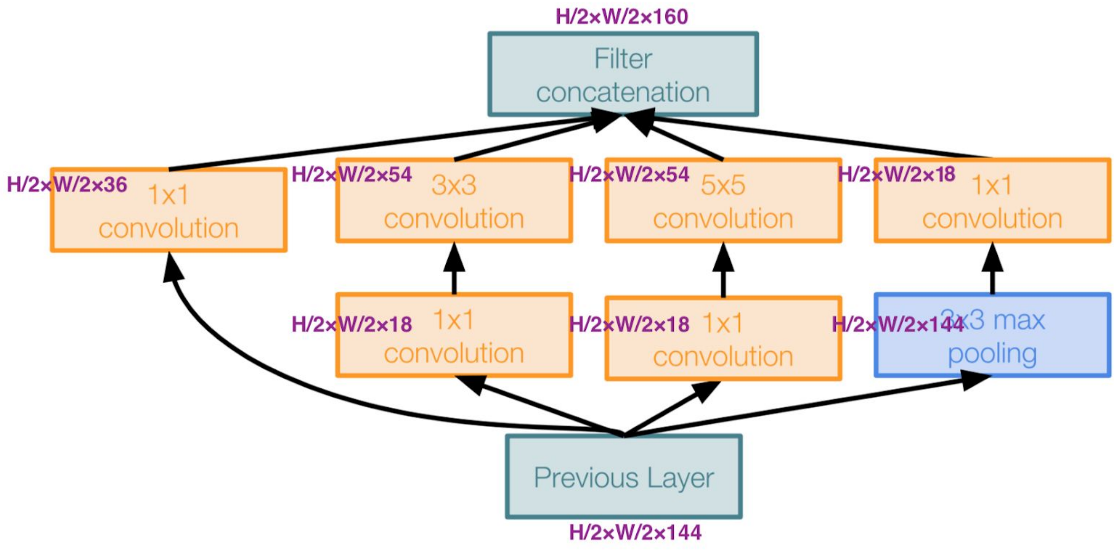

Figure 3.3 2nd inception module structure, with dimension reduction

In our encoder, we first add the feature extractor part which consists of a convolution layer and two inception modules. The output of the feature extractor has dimension of H/4×W/4×160. Next, we add a max pooling layer and three convolution layers with decreasing depth, which plays the role of feature selector.

Type	Filter size / stride	Depth	Output size	Padding	Activation

input			H×W×3		
convolution	5×5 / 1	64	H×W×64	same	relu
max pooling			H/2×W/2×64	same	
1st inception			H/2×W/2×144		
2nd inception			H/2×W/2×160		
max pooling			H/4×W/4×160	same	
convolution	3×3 / 1	45	H/4×W/4×45	same	relu
convolution	3×3 / 1	15	H/4×W/4×15	same	relu
convolution	3×3 / 1	compressed layer depth	H/4×W/4×compressed layer depth	same	relu

Table 3.2 detailed information of the simplified GoogLeNet encoder

D. Compression ratio

One of our goals is to compare the models under different compression ratios. However, the compression ratio depends on many factors, and the focus of this task is not to conduct thorough and rigorous experiments on each of the factors. Therefore, for the sake of simplicity, we change the compression ratio by adding or removing nodes of the last encoder layer. The number of nodes is represented by compressed layer depth in the Table 3.2. An additional advantage of such design is to preserve the overall network structure, which serves our goal of comparing the performances of different network structures.

E. The decoder

The decoder is the reverse of AlexNet-like encoder, where we use stacked convolution layers for feature selection and upsampling layers to change the spatial dimensions.

Type	Filter size / stride	Depth	Output size	Padding	Activation
convolution	1×1 / 1	compressed layer depth	H/4×W/4×depth	same	relu
convolution	3×3 / 1	16	H/4×W/4×16	same	relu
convolution	3×3 / 1	32	H/4×W/4×32	same	relu

convolution	1×1 / 1	64	H/4×W/4×64	same	relu
upsampling			H/2×W/2×64	same	
convolution	5×5 / 1	128	H/2×W/2×128	same	relu
upsampling			H×W×128	same	
convolution	7×7 / 1	3	H×W×3	same	linear

Table 3.3 detailed information of the decoder

- **Experimental Studies**

A. Dataset Description:

For both two networks, we trained them with the same frames from all three datasets: Basketball Drill, Blowing Bubbles, and Race Horse. The train frames were generated by collecting the first 3 frames of every 10 frames in each dataset. For example, for the first 10 frames in Basketball Drill, the 0-2 frames are selected as training frames. The initiative after such a mechanism is to guarantee the consistency of training data while making sure the data are equally distributed in each dataset. In total, 390 frames were selected from three datasets for training.

After training frames were selected, we divide each of them into small blocks with the dimension of 32x24 pixels. By doing this, we were able to train our networks with frames from all three datasets together despite their different frame dimensions and improve our training speed.

The testing set was simply the rest of frames in the datasets, which in total has 910 frames. The testing frames were also divided into small blocks with the same fashion as the training frames.

B. Quantitative evaluation:

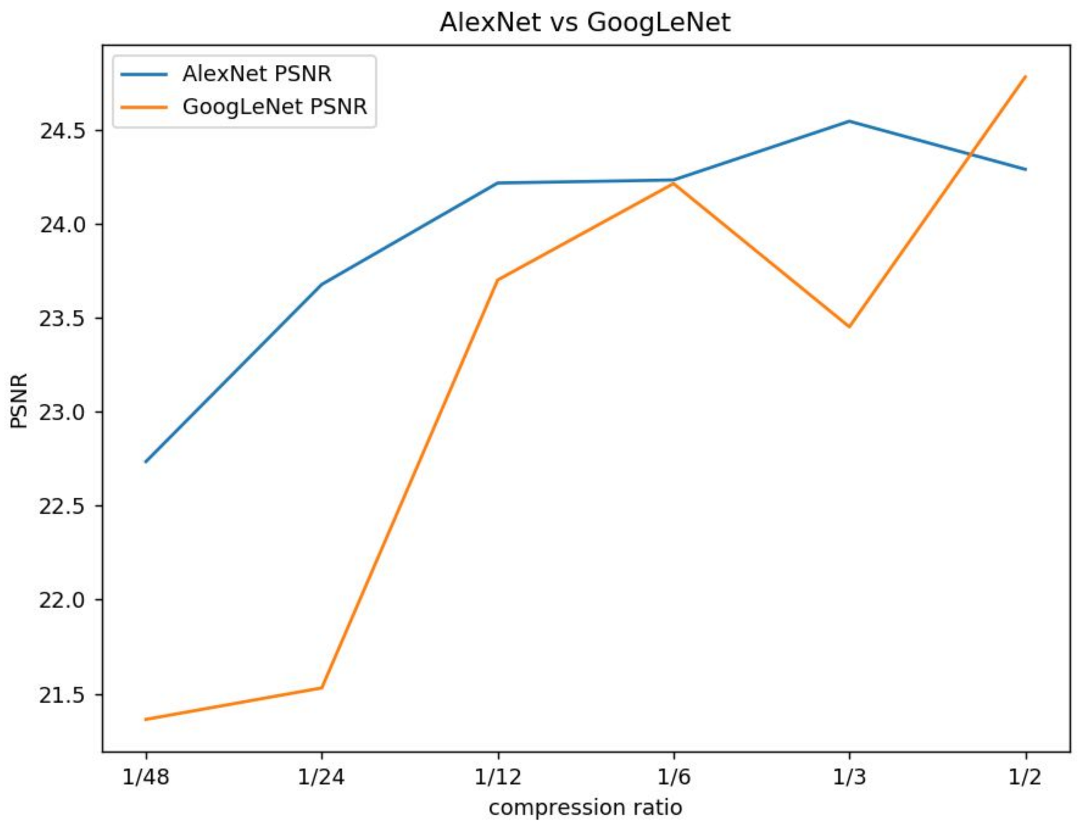

Figure 3.4 PSNR of two models

From figure 3.4, a positive correlation can be observed between the compression ratio and PSNR, though there do exist a few exceptions. The observation is intuitive since a larger compression ratio means more information can be preserved from the original image during the compression process, which also indicates better accuracy of the decompressed output.

Despite the positive correlation between the compression ratio and PSNR, a larger compression ratio will also result in larger sizes of the compressed images. Considering the goal of image compression is to reduce the data load for transmission, we should also prevent the size of compressed images from getting too large. Therefore, we could argue that higher decompression accuracy comes at the cost of higher transmission data load. In other words, the video compression task is a trade-off between reconstruction quality and compression ratio.

C. Perceptual quality evaluation:

From figure 3.5 and 3.6, a positive correlation can be observed between the visual quality and PSNR. Such observation is very intuitive because a higher PSNR means the color encoding of our decompressed image is closer to color encoding of the original image. Therefore, a model with higher PSNR is more likely to produce decompressed images with better visual quality.

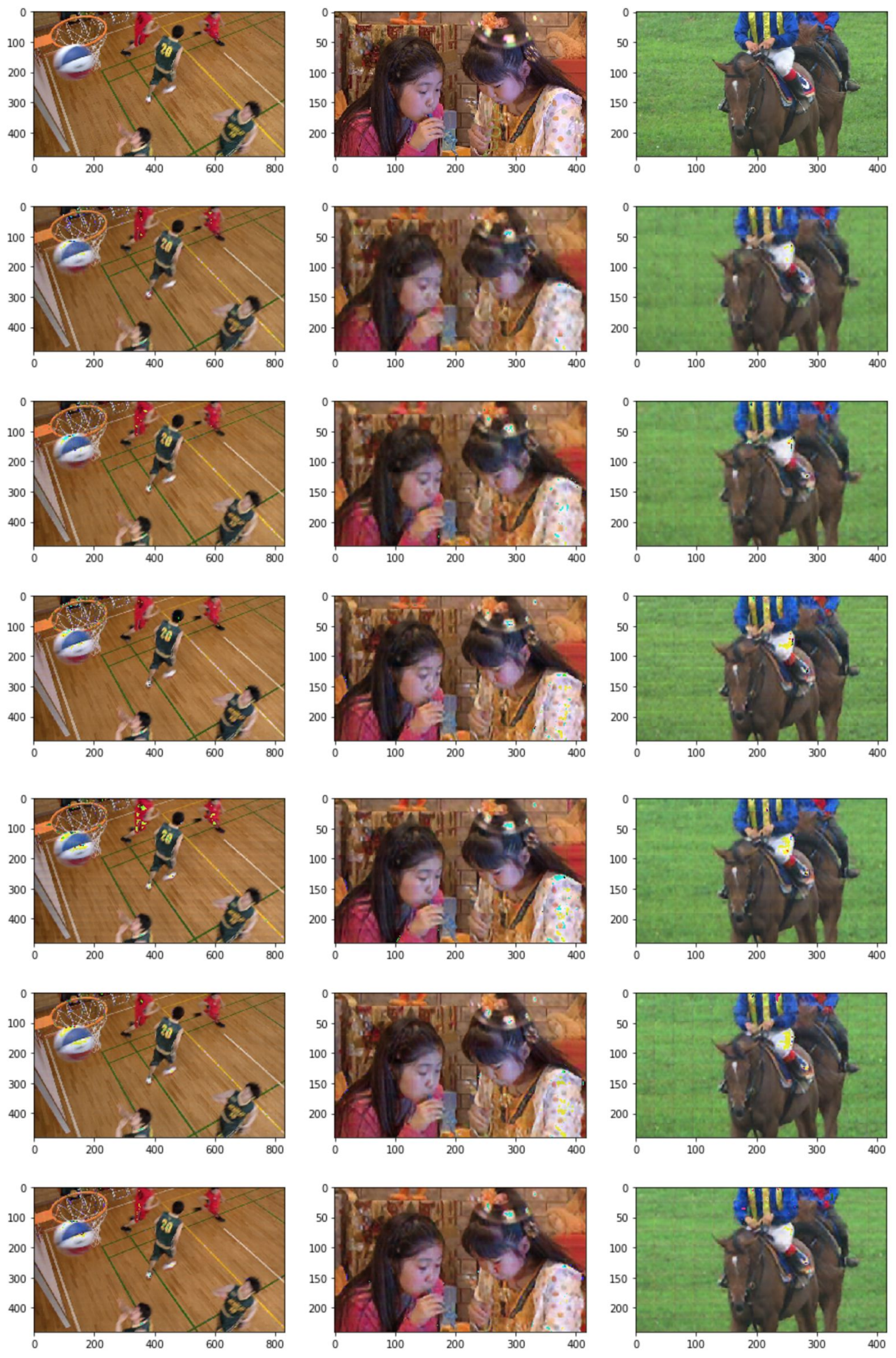

Figure 3.5 Original images (1st row) and AlexNet-like model decompressed images

when PSNR = 22.74, 23.68, 24.22, 24.23, 24.55, 24.29 (2nd to 7th row)

0 100 200 300 400
0 200 400 600 800
0 50 100 150 200
0 100 200 300 400

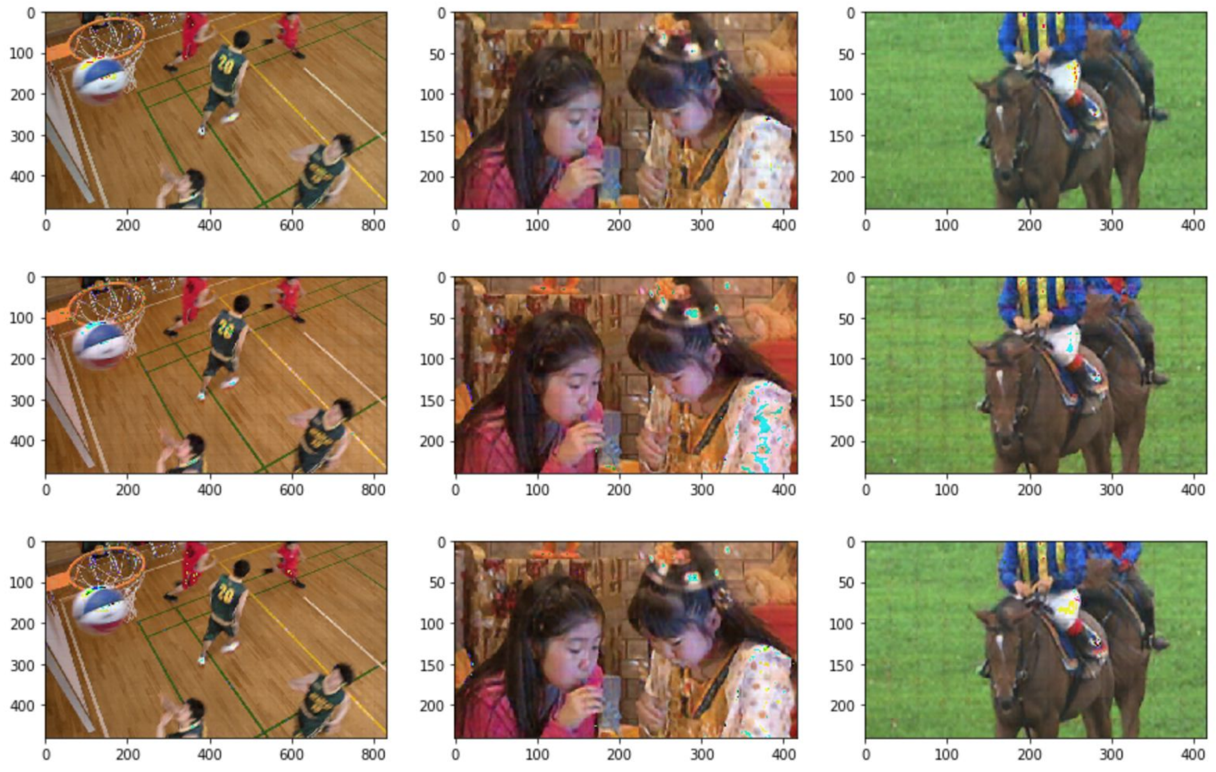

Figure 3.6 Original images (1st row) and simplified GoogLeNet model decompressed images when PSNR = 21.36, 21.53, 23.70, 24.22, 23.45, 24.78 (2nd to 7th row)

D. Complexity and model size analysis:

a. Number of Parameters

The equation to calculate the number of parameters for each layer:

$$\# parameters = filterW \times filterH \times filterD \times InputD$$

Parameters for decoder:

Type	Filter size / stride	Depth	Output size	Parameter#
convolution	1×1 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×depth	1,4,16,64,256,576
convolution	3×3 / 1	16	H/4×W/4×16	144,288,576,1152,2304,3456
convolution	3×3 / 1	32	H/4×W/4×32	4,608
convolution	1×1 / 1	64	H/4×W/4×64	2,048
upsampling			H/2×W/2×64	
convolution	5×5 / 1	128	H/2×W/2×128	204,800
upsampling			H×W×128	
convolution	7×7 / 1	3	H×W×3	18,816
		compressed layer depth d = 1,2,4,8,16,24		230417, 230564, 230864, 231488, 232832, 234304

Parameters for model #1 encoder:

Type	Filter size / stride	Depth	Output size	Parameter #
input			H×W×3	
convolution	7×7 / 1	128	H×W×128	18,816
max pooling			H/2×W/2×128	

convolution	5×5 / 1	64	H/2×W/2×64	204,800
max pooling			H/4×W/4×64	
convolution	1×1 / 1	32	H/4×W/4×32	2,048
convolution	3×3 / 1	16	H/4×W/4×16	4,608
convolution	3×3 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×compressed layer depth	144,288,576,1152,2304,3456

Therefore, the total parameters of model#1 for each d values are:

Compresse layer depth	1	2	4	8	16	24
Total Parameters	460,833	461,124	461,712	462,912	465,408	468,032

Parameters for model #2 encoder:

Type	Filter size / stride	Depth	Output size	Parameter #
input			H×W×3	
convolution	5×5 / 1	64	H×W×64	4,800
max pooling			H/2×W/2×64	
1st inception			H/2×W/2×144	55,808
2nd inception			H/2×W/2×160	110,808
max pooling			H/4×W/4×160	
convolution	3×3 / 1	45	H/4×W/4×45	64,800
convolution	3×3 / 1	15	H/4×W/4×15	6,075
convolution	3×3 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×compressed layer depth	135,270,540,1080,2160,3240

Therefore, the total parameters of model#2 for each d values are:

Compresse layer depth	1	2	4	8	16	24
Total Parameters	472,843	473,125	473,695	474,859	478,775	479,835

b. Computational complexity

The equation to calculate the complexity for each convolution layer:

complexity = outputW x outputH x outputD x filterW x filterH x inputD

Complexity for decoder:

Type	Filter size / stride	Depth	Output size	Complexity
convolution	1×1 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×depth	48,192,768,3072,12288,27648
convolution	3×3 / 1	16	H/4×W/4×16	6912,13824,27648,55296,110592,165888
convolution	3×3 / 1	32	H/4×W/4×32	221,184
convolution	1×1 / 1	64	H/4×W/4×64	98,304
upsampling			H/2×W/2×64	
convolution	5×5 / 1	128	H/2×W/2×128	39,321,600
upsampling			H×W×128	
convolution	7×7 / 1	3	H×W×3	18,816
		compressed layer depth d = 1,2,4,8,16,24		39666864, 39673920, 39688302, 39718272, 39782784, 39853440

Complexity for model #1 encoder:

Type	Filter size / stride	Depth	Output size	Complexity
input			H×W×3	
convolution	7×7 / 1	128	H×W×128	14,450,688
max pooling			H/2×W/2×128	
convolution	5×5 / 1	64	H/2×W/2×64	9830400
max pooling			H/4×W/4×64	
convolution	1×1 / 1	32	H/4×W/4×32	98304
convolution	3×3 / 1	16	H/4×W/4×16	221184
convolution	3×3 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×compressed layer depth	6912,13824 ,27648,552 96,110592, 165888

Therefore, the total complexity of model #1 for each d values are:

Compresse layer depth	1	2	4	8	16	24
Total Parameters	~64,270 ,000	~64,280 ,000	~64,300 ,000	~643,700 ,000	~644,900 ,000	~646,200 ,000

Parameters for model #2 encoder:

Type	Filter size / stride	Depth	Output size	Complexity
input			H×W×3	
convolution	5×5 / 1	64	H×W×64	3,686,400
max pooling			H/2×W/2×64	
1st inception			H/2×W/2×144	5,996,544
2nd inception			H/2×W/2×160	8,833,536
max pooling			H/4×W/4×160	
convolution	3×3 / 1	45	H/4×W/4×45	3,110,400
convolution	3×3 / 1	15	H/4×W/4×15	291,600

convolution	3×3 / 1	compressed layer depth d = 1,2,4,8,16,24	H/4×W/4×compressed layer depth	6480,12960, 25920,51840,103680,155520

Therefore, the total complexity of model #2 for each d values are:

Compresse layer depth	1	2	4	8	16	24
Total Parameters	~61,590,000	~61,600,000	~61,630,000	~616,870,000	~618,000,000	~619,330,000

- **Conclusions and Future Work**

In this task, we propose two CNN-based structures to build a color video compression system. We try to keep both models of similar sizes by controlling the total parameters. Similar computational complexity is also guaranteed by this design. Good reconstruction visual quality can be achieved by both models given a high compression ratio. While the compression ratio is low, the AlexNet-like model performs better.

Looking into the future, there are some directions that we could try to further improve the performance of our video compression system.

First, training our model with more data and more epoches. Through our experiment, we notice that increasing training data size and number of epochs can help improve the performance of the system. However, due to our limited computation power and the strict time frame, we have to limit the size of our training set and number of epochs to the current value. We hope a more comprehensive training can give us a better reconstruction quality at the same compression ratio.

Second, we could explore other parameter settings to improve the performance. Apparently, with the current network structure unchanged, there are many approaches to achieve the same compression ratio, including changing stride size, adding zero padding or changing the filter size. In this task we only one used parameter settings for all the training cases, but if we could have more time to tune the parameters, we believe that our system could have better performance.

Lastly, some techniques from recent research should be applied to our video compression system in the future. A system that considers both the inter-frame and intra-frame prediction like the DeepCoder (Tong et al. 2017) and a video encoder that using generative adversarial network (GAN) are two promising directions that are worth trying.

- **Contribution**

Both authors are equally contributed to this report.

- **References**

Chen, Tong, et al. "Deepcoder: A deep neural network based video compression." ***2017 IEEE Visual Communications and Image Processing (VCIP)*****. IEEE, 2017.**

Szegedy, Christian, et al. "Going deeper with convolutions." ***Proceedings of the IEEE conference on computer vision and pattern recognition*****. 2015.**