Performance Evaluation of a Combined Convolutional AutoEncoder and Image Recognition Model for Large Scale Images

Takuho Myojin[1][0009−0000−1759−5539] and
Yukinobu Hoshino[2][0000−0001−5285−8007]

185 Miyanokuchi, Tosayamada, Kami City, Kochi 782-8502, JAPAN

Abstract. There is a need to introduce remote disaster monitoring camera systems that utilize AI technology. This is especially the point where disasters have become larger and more frequent in recent years. The challenge to realize this system is to acquire images with cameras and to transmit this information stably and quickly. I then turned my attention to the CAE(Convolutional AutoEncoder). The reason for using CAE was to reduce the amount of data compared to images captured by cameras and to improve security. In this paper, we use this CAE to design an AI model that integrates an image recognition model for image restoration verification and disaster detection, and verify its performance.

Keywords: AutoEncoder · CNN · Deconvolution · AI

1 Introduction

In recent years, natural disasters have occurred frequently around the world. In particular, damage from external flooding in dams and rivers has been occurring on a large scale around the world. As a countermeasure, it is necessary to provide quick and stable information in the event of a disaster. In doing so, it is necessary to introduce a surveillance camera system with high security built in. Furthermore, the low communication speed environment in mountainous areas and the installation of the system must be met. Our laboratory has been successful in using AI to predict floods and heavy rainfall in reservoir and river areas in Sri Lanka[1, 2], and the reason for the success of AI prediction is the use of a neural network in images called a CNN (Convolutional Neural Network). CNN[3] is an innovative technology in the field of machine learning and computer vision. It had achieved remarkable results, especially in tasks such as image recognition, object detection, and face recognition. In addition, image size can be compressed by setting the necessary parameters. Furthermore, they have also focused on implementing neural networks such as CNNs in FPGAs; by implementing them in FPGAs, AI technology can be incorporated into IoT devices. By integrating these technologies, it is possible to implement the AI technology surveillance camera system mentioned earlier.

1.1 Research Purpose

Based on the results of the laboratory, this paper aims to develop and validate an AI image compression technique using CNN for a remote surveillance camera system for disaster prevention monitoring, as shown in Figure 1. AI image compression can compress the size of the image while maintaining the necessary information in the image and transmit it. In addition, image compression reduces the amount of information required for communication, allowing for quick and stable transmission. The next step is to use AI magnification techniques to maintain the original image and classify whether a disaster has occurred. In this paper, we experimented with Figure 1 once all on software. In doing so, in the AI image compression and restoration, the experiment was conducted to see if the image could be compressed until it could be restored, as well as to take into account the accuracy of the image classification.

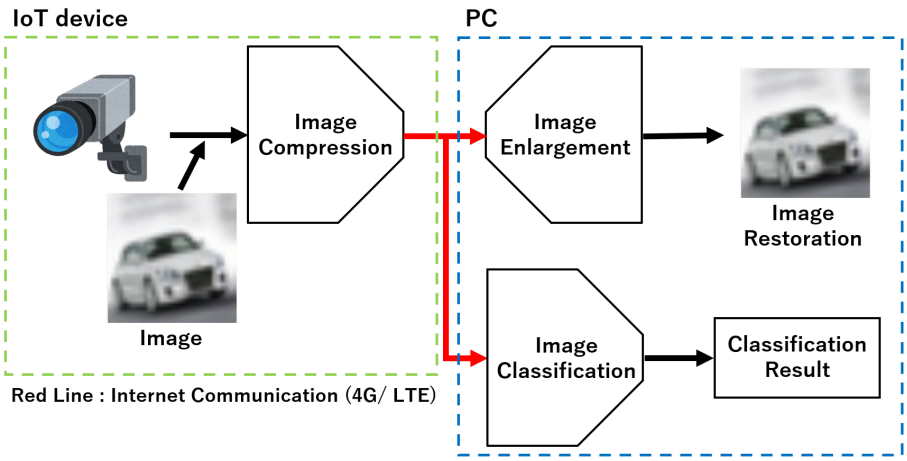

Fig. 1. Overall Process

1.2 Structure of this paper

The contents of this paper are as follows. Chapter 2 describes the necessary AI technology and image processing used for the AI model. Chapter 3 describes the details of the experiment. Chapter 4 describes the experimental results and discussion. After that, we discuss future research topics and prospects.

2 About the AI technology used

This chapter describes the basic technology of the AI model used in the experiment.

2.1 CNN(Convolutional Neural Network)

Convolutional Neural Network (CNN) is a technique first published in the literature [3]. Before the convolutional layer was proposed, image processing was done by a person deciding the information extraction filters. Convolutional layers are filters that extract information by filter, stride and padding. The advantage of the convolution layer is that spatially close pixels have similar values and features are extracted by loosely coupling while sharing parameters without erasing image characteristics that are closely related between each RGB channel. Loosely coupling allows, in particular, to ignore the relationship between distant pixels. Below is an equation showing the change in output size of the CNN.

2.2 Output size of CNN

The output size $H_o \times W_o$ of the convolution layer is given by Equation (1) when the image input size is $H_i \times W_i$, filter size is $k_h \times k_w$, stride is $s_h \times s_w$, and padding is $p_h \times p_w$. The output size can be adjusted by adjusting the filter size, stride and padding.

$$H_{\mathrm{o}} = \left\lfloor \frac{H_{\mathrm{i}} + 2p_h - k_h}{s_h} \right\rfloor + 1 \quad W_{\mathrm{o}} = \left\lfloor \frac{W_{\mathrm{i}} + 2p_w - k_w}{s_w} \right\rfloor + 1 \tag{1}$$

2.3 Residual Block

Residual Block is a specific structure of ResNet from the literature [4] that was noted for its ability to effectively train very deep networks. While ordinary deep networks are prone to gradient loss and gradient explosion problems as more layers are added, these problems can be mitigated by introducing ResNet's Residual Block. In a residual block, there is a shortcut path (skip connection) as shown in the Figure 2, in which the input is directly added to the output. This structure allows the model to skip layers and propagate information, facilitating the learning of deep networks.

2.4 Deconvolution

Deconvolution is used in the literature [5] to restore an image whose dimensions have been compressed by convolution to its original input size by the inverse operation of a convolutional neural network. Deconvolution is important when deconvolution is needed at each layer in the network because it performs the reverse of the convolution operation within the convolutional neural network. This is the technique used in image restoration and image generation, such as AutoEncoder and GANs (Generative Adversarial Networks).

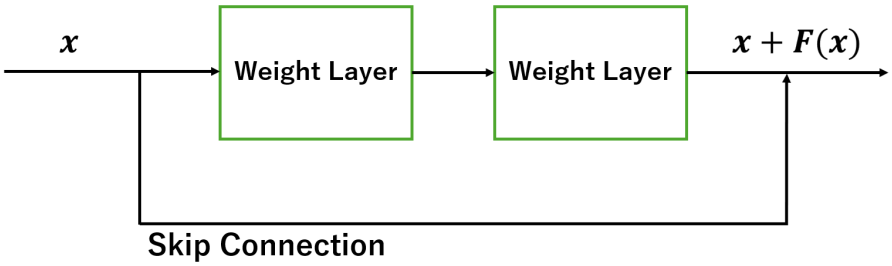

Fig. 2. Residual Block Process

2.5 AutoEncoder

AutoEncoder is an image restoration proposed in the literature [6], which reduces the dimensionality by forming a number of layers, one layer being a stack-trained Boltzmann machine with a single feature detector. Then, a configuration was proposed to expand the Boltzmann machine to restore the reduced dimensionality. In general, AutoEncoder is composed of Encoder and Decoder parts as shown in : Figure 3. In Encoder, the dimension of the input data is gradually reduced; in Decoder, the gradually reduced dimension is returned to the input dimension. Therefore, AutoEncoder is a neural network architecture that inputs an image to Encoder, compresses the dimension, and then restores the image back to the input image in Decoder. In this study, we focused on convolutional AutoEncoder, which adapts CNN for Encoder and Deconvolution for Decoder. Convolutional AutoEncoder is essential for learning biologically plausible features. Initializing the CNN with filters from the trained CAE stack yields superior performance on the number recognition (MNIST) and object recognition (CIFAR10) benchmarks[7].

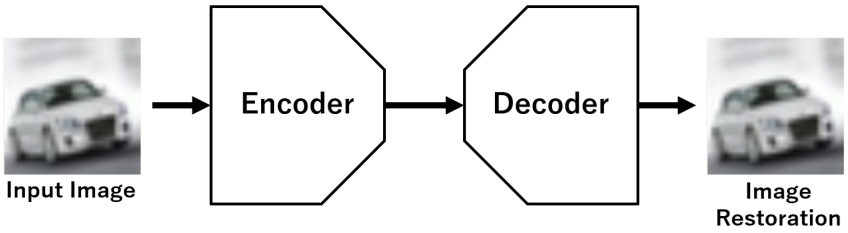

Fig. 3. AutoEncoder Process

3 Experimental Details

The dataset used in the experiment was a CIFAR10 dataset unrelated to the disaster, compressed into three classes (cars, cats, and birds) and the size of one image was increased from 32 x 32 to 320 x 320. The reason is that the images can be compressed more by using existing datasets to focus on AutoEncoder restored images and by increasing the image size as much as possible. To reproduce the same model as in the figure 1, we created an image restoration model using AutoEncoder and an image recognition model using Residual Block. The structure of these models is described in Tables 1, 2, 3, and 4. the first model created is in Tables 1 and 2. the second model is in Tables 3 and 4. Conv stands for CNN and describes the stride and kernel size at that time. Then, the data size after the processing of Layer and what kind of processing was done after the CNN. function is described. Similarly, Pool, Deconv, and Upsampling are shown as well.

Table 1 shows the structure of the AutoEncoder for the first model: in the Encoder, Conv1 doubles the number of channels by CNN, but halves the image size in height and width. In the Decoder, Upsampling Bilinear increases the image size by 2 pixels in each direction. Finally, the image is enlarged to the original size by Deconvolution(DeConv1). The evaluation function of AutoEncoder is shown in Equation (2) below. In equation (2), the cross-entropy error between the input and output images is measured for each channel and the average is calculated.

Before showing the structure of the image recognition model, the structure of the Residual Block is shown in Table 5. In Table 5, the number of channels is doubled in Conv1 and Conv3. In Conv2, the number of channels is doubled and the image size is reduced to half. In Skip Connection, the number of channels is increased by a factor of 4 and the image size is reduced by half. However, this was only activated when the stride was 2 x 2. Table 2 shows the image recognition structure of the first model: with the output of Encoder as input, the number of channels was changed by Conv1 and Pool1 to 8, a power of 2, and the image size was changed to 100 x 100. Then, 16 layers of Residual Block were designed. Finally, we compressed the image to the number of classes by Linear of all the joins.

In Table 3, the structure of AutoEncoder for the second model is shown. Initially, Conv1 was set to halve the image size horizontally and vertically and double the number of channels. Then, Pool1 reduces the image size by 2 pixels in height and width. Conv2 also doubles the number of channels in the CNN, but the image size is reduced to half the height and width. In Decoder, Upsampling Bilinear increases the image size by 2 pixels in each direction, and finally Deconvolution increases the image size by a factor of 2. Then, Upsampling Bilinear (Upsampling Bilinear2) and DeConvolution (DeConv2) were combined once again to enlarge the image to its original size. As in Table 1, the evaluation function for AutoEncoder was set to Equation (2) below. Table 4 shows the image recognition structure of the first model: with the output of Encoder as input, the number of channels was changed by Conv1 and Pool1 to 16, a power of 2, and

the image size was changed to 40 x 40. Then, 16 layers of Residual Block were designed. Finally, we compressed the image to the number of classes by Linear of all the joins.

AutoEncoder measured Train Loss and Test Loss. Then, the output and input images were checked and the similarity of each channel was measured. The formula for the similarity of each channel is Equation (3) below. Equation (3) shows the formula for the similarity in each channel between the restored image and the input image. For the image recognition model, we measured Train Loss, Test Loss, Train Accuracy, Test Accuracy, and the actual correct response rate.

$$Loss_{mean} = -\frac{1}{NHWC}\sum_{i,j,k,l}\{x_{i,j,k,l}\log(y_{i,j,k,l})+(1-x_{i,j,k,l})\log(1-y_{i,j,k,l})\} \quad (2)$$

$$similarity_i = \frac{\mathbf{image}_{1i}\cdot\mathbf{image}_{2i}}{\|\mathbf{image}_{1i}\|\|\mathbf{image}_{2i}\|} \quad (3)$$

Table 1. AutoEncoder Design 1

Layer	Kernel	Stride	Output	Function
Input	-	-	(15, 3, 320, 320)	-
Conv1	(3, 3)	(2, 2)	(15, 6, 160, 160)	ReLU
Pool1	(4, 4)	(1, 1)	(15, 6, 158, 158)	Max, Instance Norm
Upsampling Bilinear1	(3, 3)	(2, 2)	(15, 6, 160, 160)	ReLU
DeConv1	(3, 3)	(2, 2)	(15, 3, 320, 320)	Max, Instance Norm
Output	-	-	(15, 3, 320, 320)	-

4 Experimental Results and Discussion

4.1 First model Result and Discussion

Initially, the AutoEncoder results for the first model are shown in Figure 4 and Table 1. Next, Figure 4 compares the input and output images. The rightmost image is the 320 x 320 input image, the center image is the Encoder output image (158 x 158), and the leftmost image is the restored image. The restored image shows that the color image was not restored. However, it was confirmed that the shape of the target object in the image was captured even in the grayscale image. Finally, Table 6 shows the similarity between the input and output images for each channel, and all similarities are greater than about 0.80. Therefore, despite the high similarity, the black-and-white image was recovered because the values of each channel at the same position in the image were very close. This is because the nature of images is such that if the values of each RGB channel are close, the image will be grayscale. Therefore, we believe that the AutoEncoder we designed

Table 2. Classification Design 1

Layer	Kernel	Stride	Output	Function
Input	-	-	(15, 6, 158, 158)	-
Conv1	(3, 3)	(1, 1)	(15, 8, 158, 158)	ReLU
Pool1	(59, 59)	(1, 1)	(15, 8, 100, 100)	Max, Batch Norm
Residual Block1	-	(2, 2)	(15, 32, 50, 50)	-
Residual Block2	-	(1, 1)	(15, 32, 50, 50)	-
Residual Block3	-	(1, 1)	(15, 32, 50, 50)	-
Residual Block4	-	(2, 2)	(15, 128, 25, 25)	-
Residual Block5	-	(1, 1)	(15, 128, 25, 25)	-
Residual Block6	-	(1, 1)	(15, 128, 25, 25)	-
Residual Block7	-	(1, 1)	(15, 128, 25, 25)	-
Residual Block8	-	(2, 2)	(15, 512, 12, 12)	-
Residual Block9	-	(1, 1)	(15, 512, 12, 12)	-
Residual Block10	-	(1, 1)	(15, 512, 12, 12)	-
Residual Block11	-	(1, 1)	(15, 512, 12, 12)	-
Residual Block12	-	(1, 1)	(15, 512, 12, 12)	-
Residual Block13	-	(1, 1)	(15, 512, 12, 12)	-
Residual Block14	-	(2, 2)	(15, 2048, 6, 6)	-
Residual Block15	-	(1, 1)	(15, 2048, 6, 6)	-
Residual Block16	-	(1, 1)	(15, 2048, 6, 6)	-
Linear1	-	-	(15, 73728)	-
Linear2	-	-	(15, 3)	-
Output	-	-	(15, 3)	-

Table 3. AutoEncoder Design 2

Layer	Kernel	Stride	Output	Function
Input	-	-	(15, 3, 320, 320)	-
Conv1	(3, 3)	(2, 2)	(15, 6, 160, 160)	ReLU
Pool1	(4, 4)	(1, 1)	(15, 6, 158, 158)	Max, Instance Norm
Conv2	(3, 3)	(2, 2)	(15, 12, 79, 79)	ReLU
Pool2	(4, 4)	(1, 1)	(15, 12, 77, 77)	Max, Instance Norm
Upsampling1	-	-	(15, 12, 79, 79)	ReLU
Deconv1	(3, 3)	(2, 2)	(15, 6, 158, 158)	Instance Norm
Upsampling2	-	-	(15, 6, 160, 160)	ReLU
DeConv2	(3, 3)	(2, 2)	(15, 3, 320, 320)	Instance Norm
Output	-	-	(15, 3, 320, 320)	-

Table 4. Classification Design 2

Layer	Kernel	Stride	Output	Function
Input	-	-	(15, 12, 77, 77)	-
Conv1	(3, 3)	(1, 1)	(15, 16, 77, 77)	ReLU
Pool1	(38, 38)	(1, 1)	(15, 16, 40, 40)	Max, Batch Norm
Residual Block1	-	(2, 2)	(15, 64, 20, 20)	-
Residual Block2	-	(1, 1)	(15, 64, 20, 20)	-
Residual Block3	-	(1, 1)	(15, 64, 20, 20)	-
Residual Block4	-	(2, 2)	(15, 256, 10, 10)	-
Residual Block5	-	(1, 1)	(15, 256, 10, 10)	-
Residual Block6	-	(1, 1)	(15, 256, 10, 10)	-
Residual Block7	-	(1, 1)	(15, 256, 10, 10)	-
Residual Block8	-	(2, 2)	(15, 1024, 5, 5)	-
Residual Block9	-	(1, 1)	(15, 1024, 5, 5)	-
Residual Block10	-	(1, 1)	(15, 1024, 5, 5)	-
Residual Block11	-	(1, 1)	(15, 1024, 5, 5)	-
Residual Block12	-	(1, 1)	(15, 1024, 5, 5)	-
Residual Block13	-	(1, 1)	(15, 1024, 5, 5)	-
Residual Block14	-	(2, 2)	(15, 4096, 2, 2)	-
Residual Block15	-	(1, 1)	(15, 4096, 2, 2)	-
Residual Block16	-	(1, 1)	(15, 4096, 2, 2)	-
Linear1	-	-	(15, 16384)	-
Linear2	-	-	(15, 3)	-
Output	-	-	(15, 3)	-

Table 5. Residual Block Design

Layer	Kernel	Stride	Output	Function
Input	-	-	(N, C, H, W)	-
Conv1	(1, 1)	(1, 1)	(N, 2C, H, W)	-
Batch Norm1	-	-	(N, 2C, H, W)	ReLU
Conv2	(1, 1)	(s, s)	(N, 2C, H / s, W / s)	-
Batch Norm2	-	-	(N, 2C, H / s, W / s)	ReLU
Conv3	(1, 1)	(1, 1)	(N, 4C, H / s, W / s)	-
Batch Norm3	-	-	(N, 4C, H / s, W / s)	ReLU
Skip Conv	-	-	(N, 4C, H / s, W / s)	-
Skip Batch Norm	-	-	(N, 4C, H / s, W / s)	ReLU
Output	-	-	(N, 4C, H / s, W / s)	-

this time produces a grayscale image because the values of each channel at the same position in the image are very close.

Next, the results of image recognition for the first model are shown in Figures 5 and 6. Figure 5 shows the change in Loss[-] per learning. The vertical axis is Loss[-] and the horizontal axis is the number of training's. It was confirmed that Train Loss and Test Loss increased significantly when the number of training sessions was around the 5th training session, and then gradually decreased. The fact that Loss increases and then decreases significantly once suggests that learning is being performed appropriately. Figure 6 shows the change in Accuracy[%] per learning. The vertical axis is Accuracy[%] and the horizontal axis is the number of learning. As the number of training sessions increased, both Train Accuracy[%] and Test Accuracy[%] repeatedly decreased and increased, and finally Train Accuracy and Test Accuracy[%] were calculated to be as high as 96% and 88%, respectively. The research paper[4] also states that deepening the Residual Block layer in the image recognition model reduces the error rate and greatly improves accuracy, so I can say that the image recognition model I created and tested is highly reliable.

Table 6. First model : AutoEncoder Similarity

Similarity	Value
Average Red Similarity	0.817
Average Green Similarity	0.799
Average Blue Similarity	0.810
RGB Similarity	0.809

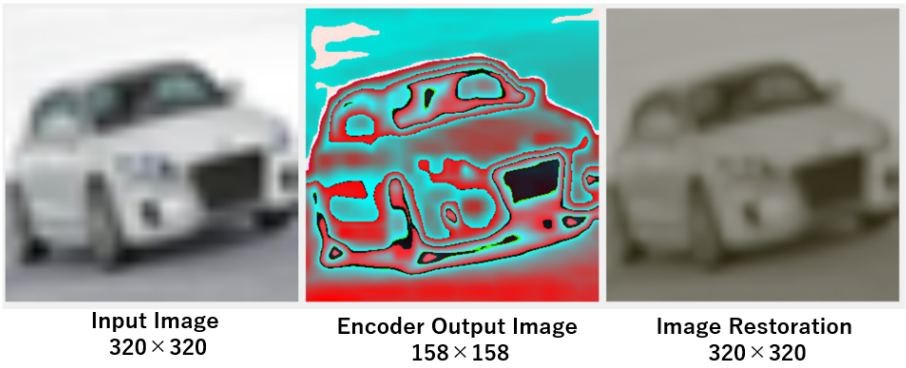

Fig. 4. First model : AutoEncoder Process

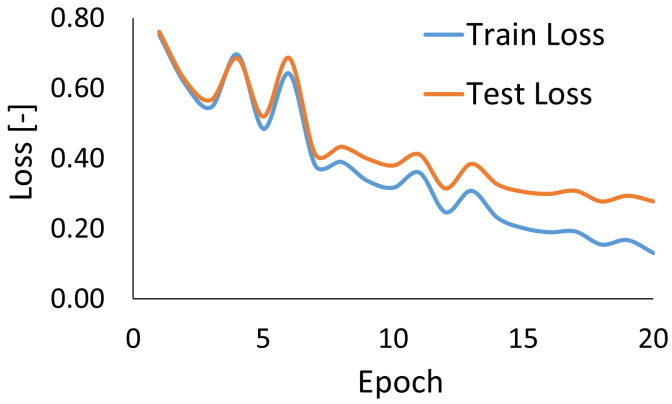

Fig. 5. First model : Image Recognition Loss Process

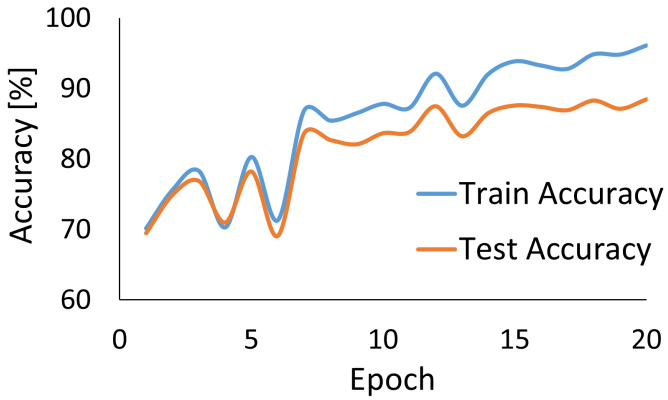

Fig. 6. First model : Image Recognition Accuracy Process

4.2 Second model Result and Discussion

The AutoEncoder results for the second model are shown in Figure 7 and Table 7. Loss confirmed that the study was successful. Figure 7 shows the input image, the Encoder output image, and the restored image. The input image is the leftmost image, the Encoder output image is the center image, and the restored image is the rightmost image. Focusing on the restored image, it was confirmed that the entire image was restored. In addition, compared to the first model, it was possible to colorize the image, but it was confirmed that it was paler than the input image. In the paper [8], it is confirmed that the model can be accurately restored using CAE, but the model created this time shows some parts where data by color is lost, so this will be an issue to be addressed in the future. Finally, Table 7 shows the similarity between the input image and the restored image, which shows a decrease in similarity compared to the image similarity of the first AutoEncoder.

The image recognition results for the second model are shown in Figures 8 and 9. Figure 8 shows the change in Loss per study. The vertical axis is Loss[-] and the horizontal axis is Epoch. Overall, a gradual decrease was observed. In addition, we could confirm the process of increasing and decreasing Loss once. Figure 9 shows the change in the percentage of correct responses per study. The vertical axis is the correct response rate [%] and the horizontal axis is Epoch. As the number of studies increased, both Train Accuracy [%] and Test Accuracy [%] repeatedly decreased and increased, and finally Train Accuracy and Test Accuracy were calculated to be 93% and 85%, respectively. However, we observed a slight drop in accuracy compared to Figure 6. Overall, we believe that a considerable accuracy can be achieved in a real environment for image recognition. Similar to the previous experiment, the paper [4] states that deepening the Residual Block layer in the image recognition model reduces the error rate and greatly improves accuracy, so the second image recognition model I created and tested is highly reliable.

Table 7. Second model : AutoEncoder Similarity

Similarity	Value
Average Red Similarity	0.966
Average Green Similarity	0.973
Average Blue Similarity	0.977
RGB Similarity	0.972

5 Summary and Prospects

Regarding the design and accuracy verification of the AI model combining image restoration and image recognition in this study, we were able to restore images to some extent using CAE in image restoration. However, we believe that more

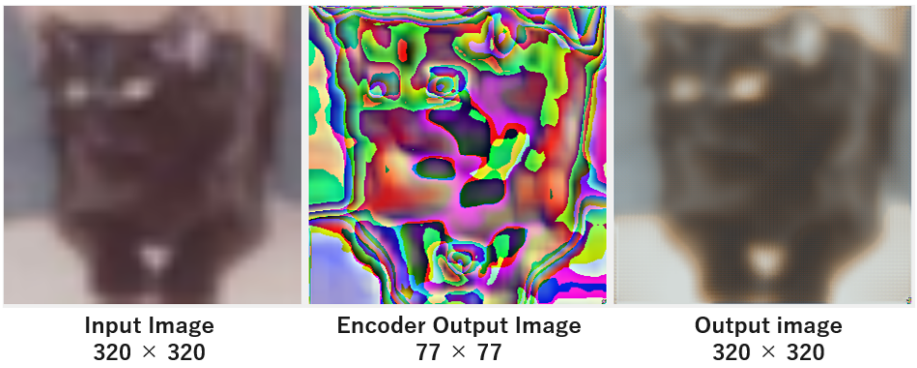

Fig. 7. Second model : AutoEncoder Process

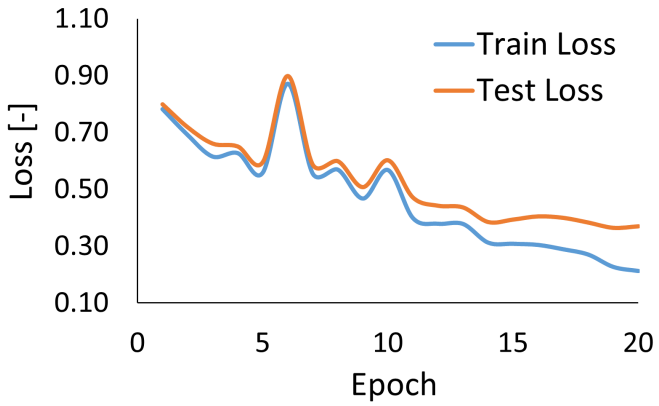

Fig. 8. Second model : Image Recognition Loss Process

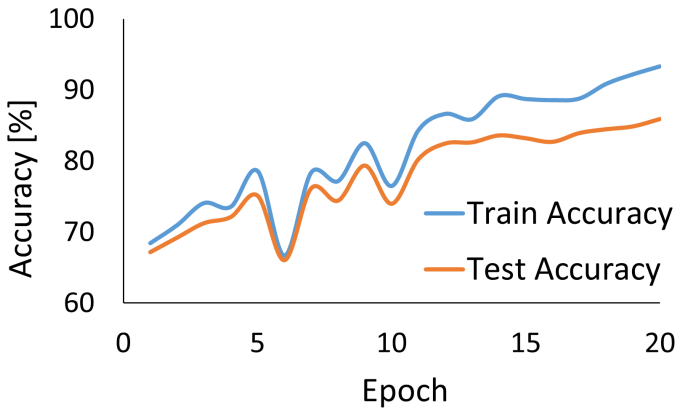

Fig. 9. Second model : Image Recognition Accuracy Process

colorization of the restored image is needed. In image recognition, high accuracy was achieved. As a future issue, it is necessary to mount the Encoder of CAE on IoT devices, and it is also necessary to consider the means of communication for this purpose.

5.1 Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 22KK0160.

References

1. Namal B. Rathnayake, Upaka Rathnayake, Dang T. Linh and Yukinobu Hoshino: Cascaded Adaptive Network-Based Fuzzy Inference System for Hydropower Forecasting, Sensors, Vol..22, No.8, 2905 MDPI(2022)
2. Madhawa Herath, Tharaka Jayathilaka, Yukinobu Hoshino, Upaka Rathnayake: Deep machine learning-based water level prediction model for Colombo flood detention area, Applied Sciences, MDPI(2023)
3. Y LeCun, L Bottou, Y Bengio, P Haffner. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
5. Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, Rob Fergus. "Deconvolutional networks." 2010 IEEE Computer Society Conference on computer vision and pattern recognition. IEEE, 2010.
6. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986." Biometrika 71 (1986): 599-607.
7. Jonathan Masci, Ueli Meier, Dan Cire¸san, and J¨urgen Schmidhuber. "Stacked convolutional auto-encoders for hierarchical feature extraction." Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21. Springer Berlin Heidelberg, 2011.
8. Zhengxue Cheng, Heming Sun, Masaru Takeuchi, Jiro Katto. "Deep convolutional autoencoder-based lossy image compression." 2018 Picture Coding Symposium (PCS). IEEE, 2018.
9. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
10. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. pmlr, 2015.