Invisible QR Code Generator Using Convolutional Neural Network

Kohei Yamauchi
Graduate School of Robotics and Design
Osaka Institute of Technology
Osaka, Japan
m1m19r36@oit.ac.jp

Hiroyuki Kobayashi
Graduate School of Robotics and Design
Osaka Institute of Technology
Osaka, Japan
hiroyuki.kobayashi@oit.ac.jp

***Abstract*—The authors aim to embed arbitrary information in arbitrary images and restore them using CNN. To achieve this goal, we propose a model consisting of two CNNs with different roles. In the proposed method, it is used as an information medium for embedding a QR code. The QR code error correction function is expected to restore the embedded information without error. Existing research has shown that embedding a QR code in a sharp color image does not restore the QR code correctly. This paper modified the CNN configuration to address this issue. The authors hope this technology can be used to integrate QR codes into human living space and hide information without upset. We learned how to embed a QR code image in a color image using the CNN model proposed this time. As a result, the authors were able to embed the QR code image without degrading the quality of the input color image. Current methods have drawbacks. Blur the image with the embedded QR code. Then, there is a problem that the embedded QR code cannot be restored. We will solve this problem in the future.**

***Index Terms*—Deep Learning, Convolutional Neural Network, Steganography**

I. Introduction

In recent years, a method using a Convolutional Neural Network (hereinafter referred to as CNN) have been successful in image classification problems. For example, in the field of image generation, models for generating images using CNN, such as GAN [1] and DCGAN [2], and algorithms for converting the style of images [3] have been proposed. Here, an algorithm for converting the style of the image will be described. When an image is input to CNN trained for object recognition, in the hidden layer, the information on the color and texture of the object in the image weakens, and the information on the shape of the object becomes stronger. Therefore, by replacing the weakened part of the information with the style information of another image, it is possible to generate an image in which only the style has changed. Now, We thought that if it was possible to replace part of the information in the input image with other arbitrary information using the above method, it would be possible to embed arbitrary information in the part of the image. We also thought that it would be possible to restore the embedded information by extracting any information embedded in the image as a feature.

Conventional methods of steganography include a method of directly changing pixel values such as luminance information and color information of an image to embed information in the image, and a method of changing frequency components (amplitude, phase) in the image [4]. In this paper, we use CNN to embed arbitrary information in a color image and to restore the embedded information, we use the QR code as a medium of the information to be embedded in the image.

Based on this idea, we have created a neural network model that embeds information by embedding a QR code in a color image using CNN [5].

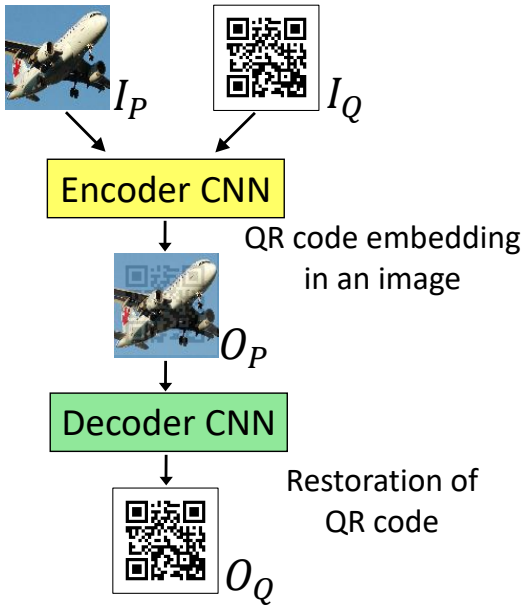

Fig. 1. Proposed CNN Model

II. Existing Method

The structure of the old method [5] is shown in Fig.1. As shown in Fig.1, the CNN embedded in the image is composed

978-1-7281-5414-5/20/$31.00 ©2020 IEEE

of two sets of CNN models with different roles. It is an Encoder that embeds a QR code image of a color image and a Decoder that restores the embedded QR code. The Encoder and Decoder configurations are shown below.

1) Encoder: The Encoder has a structure that is shown in Fig.2. Also, the details of the Encoder is shown in Fig.3. The color image I_P and the QR code image I_Q are input to the Encoder, and the embedded image O_P is output. Within the Encoder, the two inputs are individually processed by separate convolutional layers to extract image features. Next, embed the image by combining the features of the QR code image with the color image. After going through a series of processes several times, the functionality of both images is fully combined and the embedded image is output.

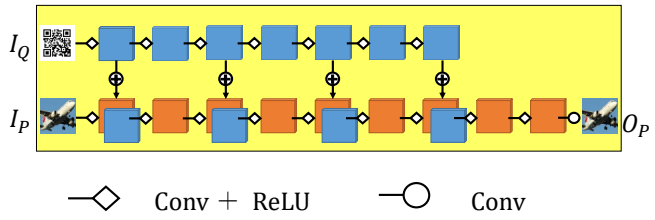

Fig. 2. Existing Encoder [5]

	Input	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv
I_h	100×100×3	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16		
		concat		concat		concat		concat	1×1×16	1×1×8	1×1×3
I_g	100×100×1	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16	3×3×16		

Fig. 3. Details of Encoder

2) Decoder: The Decoder has a structure that is shown in Fig.4. Also, the details of the Decoder is shown in Fig.5. The Decoder outputs the restored image O_Q of the embedded QR code by processing the input embedded image O_P with multiple convolutional layers.

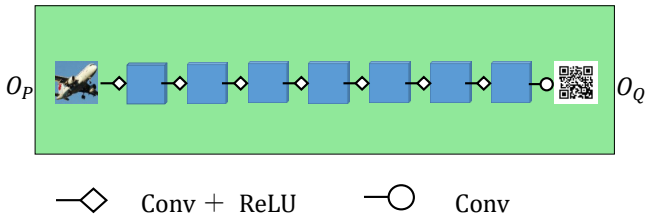

Fig. 4. Existing Decoder [5]

	Input	lambda	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv+Relu	Conv
O_e	100×100×3	100×100×1	3×3×16	3×3×16	3×3×8	3×3×8	3×3×3	3×3×3	3×3×1

Fig. 5. Details of Decoder

When training a CNN model, the two CNNs are connected and the convolutional layer weights are updated using the loss function L.

$$L(I_Q, I_P) = ||I_P - O_P||^2 + ||I_Q - O_Q||^2 + \lambda(||W_e||^2 + ||W_d||^2) \quad (1)$$

W_e and W_d represent the learned weights of the Encoder and Decoder, respectively.
I learned these two CNNs, embedded the QR code, and restored it. The images used for training I_P and I_Q are both 100x100 pixels in size, but I_Q is a 1-channel grayscale image. Finished. Fig.6 shows the result of embedding and restoring the QR code using a trained model.

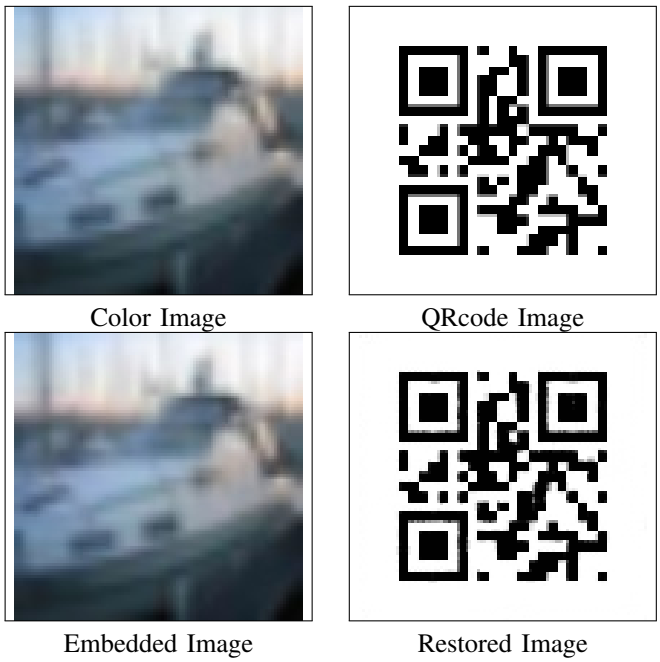

Fig. 6. Input and Output Image

The way one can embed the QR code in the image and restore it the existing way. However, there are problems with this model. The first problem is that the color image that embeds the QR code must use a blurry image like the one in Fig.6. When using an image with sharp edges, as in Fig.7, embedding and restoring the QR code fails.

The second problem is that the embedded QR code cannot be recovered without inputting the Encoder output directly to the Decoder. Image and reload the Encoder output O_e as shown in Fig.8. Input this integer data into the Decoder. In this case, the QR code embedded in the image cannot be restored.

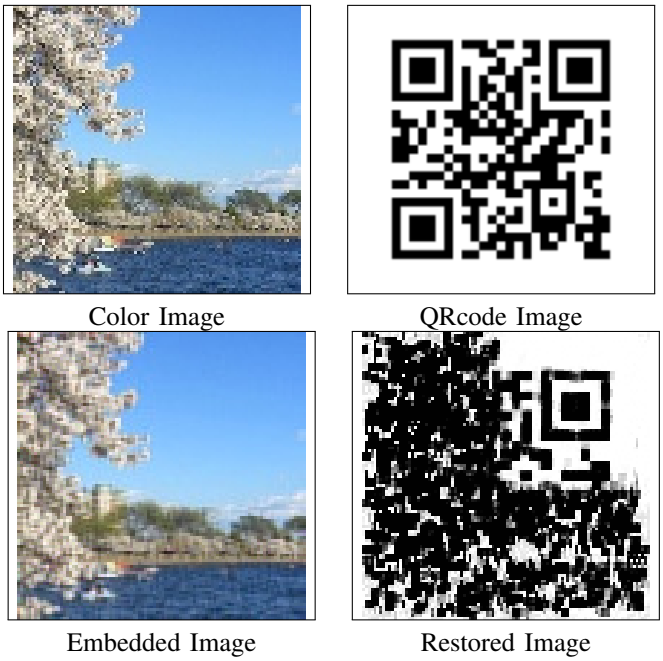

Fig. 7. Input and output images with sharp images

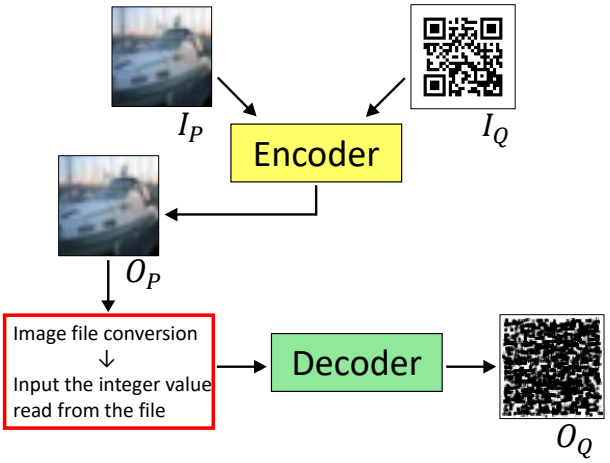

Fig. 8. Output image when the Encoder output is converted into an integer

III. PURPOSE

The purpose of this paper is to embed and restore information in images using CNN. In this paper, we modified the composition of Encoder and Decoder to solve the problem of the existing model.

The rest of this document is organized as follows: Section IV details the proposed CNN model. Section V describes the details of the experiment. Section VI shows the results of the experiment. Section VII presents insights from experimental results. Section VIII concludes this document with a summary and future work.

IV. METHOD

The proposed method is similar to the existing model in that it consists of a set of two CNN models with different roles, as shown in Fig.1. In the design, the authors changed the configurations of the Encoders and Decoders. Encoders and Decoders This time we'll talk more about Decoders.

A. Encoder

Fig.9 shows the structure of the Encoder. The color image I_P and the QR code image I_Q are input to the Encoder, and the embedded image O_P is output. This Encoder is based on the method of Xintao et al [6]. This is a neural network called tiramisu [7], proposed for image segmentation.

The model contains 11 Dense Blocks, 5 Transition Up and 5 Transition Down. The Dense Blocks, Transition Down, and Transition Up of Fig. 9 are explained below.

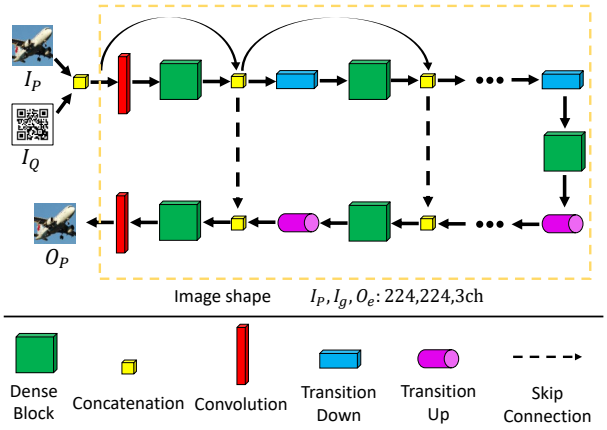

Fig. 9. Encoder

1) Dense Block: The Dense Block is configured as shown in Fig.10. Dense Block solves the gradient vanishing problem by concatenating the output processed by batch normalization, ReLU, 3x3 convolutional layers, and dropout with the input array and the existing output array. Also, the number of parameters can be significantly reduced. The dropout rate is set to 0.2. It generates a feature map using Dense Blocks.

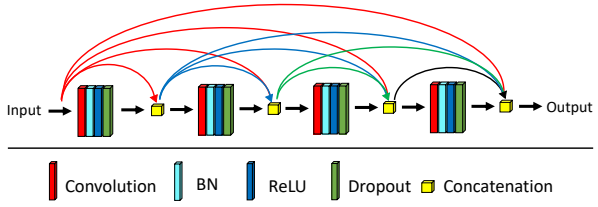

Fig. 10. Dense Block

2) Transition Down: Transition down consists of batch normalization, ReLU, and 1x1 convolutional layers. Setting the convolution stride length to 2 will reduce the spatial resolution like pooling layers.

3) Transition Up: Transition Up consists of a 3x3 transposed convolutional layer. The length of the convolution stride is set to 2 and the feature map is expanded.

B. Decoder

Fig.11 shows the structure of the Decoder. The Decoder outputs the restored image O_Q of the embedded QR code by processing the input embedded image O_P with multiple convolutional layers. This Decoder has a Batch Normalization layer added compared to the existing Decoder.

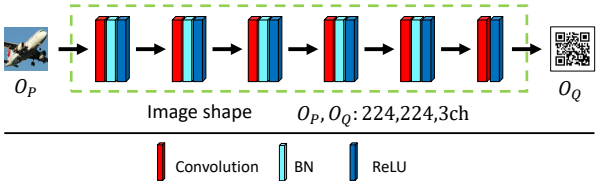

Fig. 11. Decoder

When training a CNN model, the two CNNs are connected and the convolutional layer weights are updated using the loss function L.

$$L(I_Q, I_P) = ||I_P - O_P|| + \beta||I_Q - O_Q|| \tag{2}$$

where, β is a hyperparameter that adjusts the degree of influence of the Decoder during learning.

V. Experiment

Train the proposed CNN model. After training, embed the QR code and see if you can restore the model to a crisp image. Also, compare the results with the existing model to see if the problems with the existing model are resolved.

A. Train the Proposed CNN Model

First, train the proposed CNN model. Second, input the color image I_P and the QR code image I_Q into the Encoder. Currently, the image size for both images is 224 x 224 pixels. Then the embedded image O_P is output. Then the output embedded image O_P is input to the Decoder and the restored image O_Q is output. During training, the Encoder output and Decoder input are connected as shown in Fig.12, and the model weights are updated at the same time.

As a training dataset, we used ImageNet [8], which is a dataset that Stanford University collected and classified images from the Internet. We have prepared 7200 color and QR code images for training and 2800 images for verification. Images on ImageNet come in a variety of sizes. Therefore, the images used for learning were resized to 224x224 pixels. Then, the epoch was 200, the batch size was 4, the Decoder effect during training was β 0.75, the optimization algorithm was RMSprop [9], and the learning rate was 0.001.

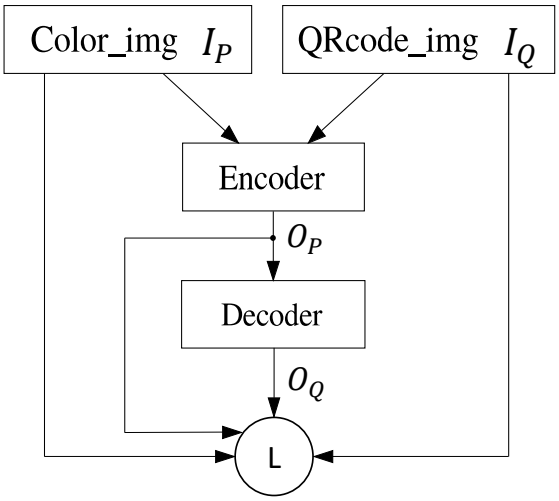

Fig. 12. CNN Model at Training

B. QR code image embedding for color image

After training, use CNN to embed the QR code image I_Q into the color image I_P. For I_P and I_Q we used images that were not used during training. First, input I_P and I_Q into the Encoder and output the embedded image O_P. Currently O_P is floating-point data. Save the Encoder output to an image file and reload that file, as shown by Fig.13. O_P is integer data. Then input the Decoder O_P and output the image O_Q with the restored QR code embedded. Then compare it to the output of the existing model.

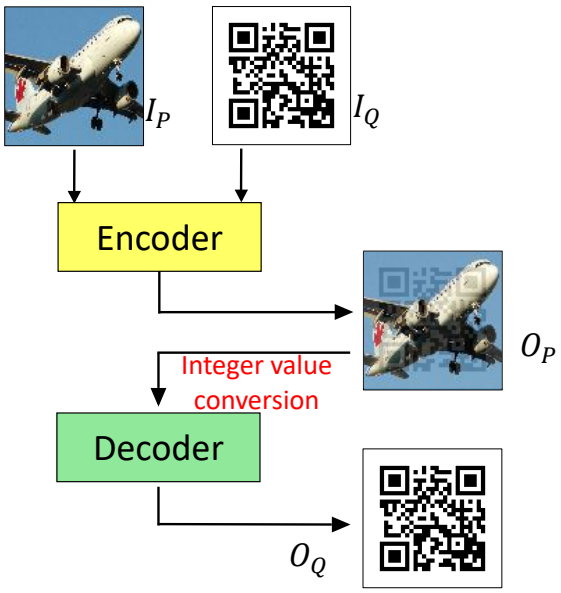

Fig. 13. Input and output overview

VI. Results

Fig.14 shows the result of embedding a QR code in a color image using the old CNN model, training the CNN model, and then restoring.

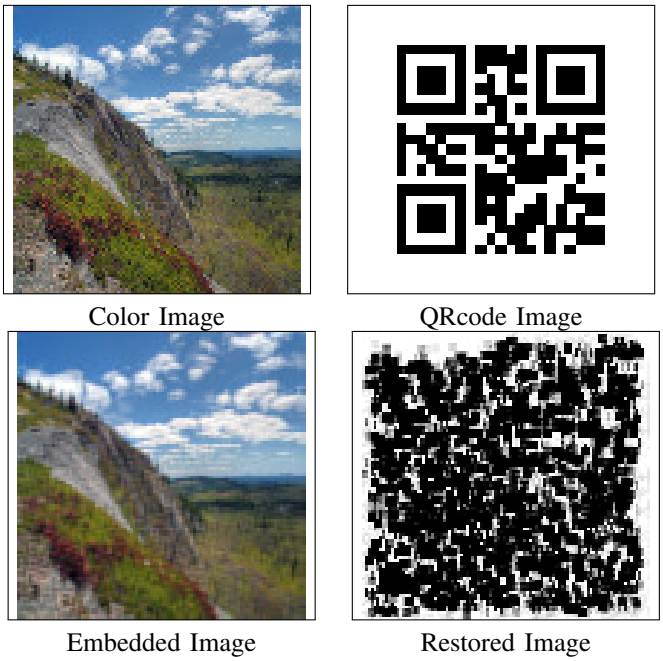

Fig. 14. Input and Output Image (existing model)

Fig.15 shows the result of embedding a QR code in a color image using the CNN model trained this time, training the CNN model, and then restoring it.

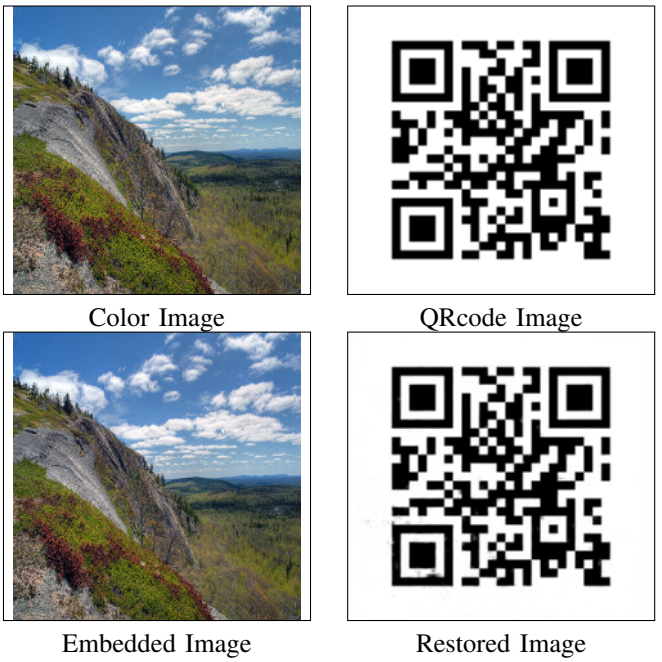

Fig. 15. Input and Output Image (proposed method)

VII. Discussion

Looking at the Fig.15, the image output by this Encoder looks similar to the input color image. Also, compared with the input QR code image, the restored image output from the Decoder has a slightly distorted image, but you can see that the input QR code can be reproduced almost.

Also, with the existing model, if you embed the QR code in an image with clear contours and then restore it, you cannot restore the embedded QR code as shown in Fig.14. Also, the embedded QR code could not be restored without inputting the Encoder output directly to the Decoder. By using the proposed method, we were able to solve these two problems.

VIII. Conclusions

A. Summary

By using the model created this time, we were able to solve the problem that the QR code could not be embedded and restored in an image with clear contours.

B. Current Issue

Currently, there is a problem that if one does blur processing on an embedded image as shown in Fig.16, it cannot be restored properly. To solve this problem, we improve the learning method and model.

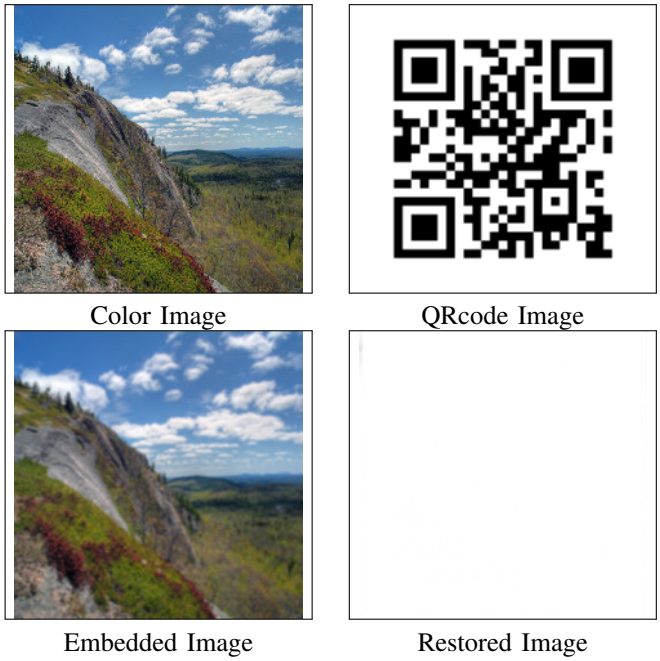

Fig. 16. Input image and output image with blurring process

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[4] R. Krenn, "Steganography and steganalysis," *Retrieved September*, vol. 8, p. 2007, 2004.
[5] K. Yamauchi and H. Kobayashi, "Embedding of qr code image for arbitrary image by cnn," in *Robotics and Mechatronics Conference Lecture Summary 2019*. The Japan Society of Mechanical Engineers, 2019, pp. 2P1–I02.
[6] D. Xintao and L. Nao, "Hide the image in fc-densenets to another image," *arXiv preprint arXiv:1910.08341*, 2019.
[7] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 11–19.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[9] G. Hinton, "RMSProp," https://www.cs.toronto.edu/ tijmen/csc321 /slides/lecture$_slides_lec6.pdf$.