Efficient Learning of Neural Networks for Image Processing

浦添, 和哉

(Degree)
博士（工学）

(Date of Degree)
2023-03-25

(Date of Publication)
2024-03-01

(Resource Type)
doctoral thesis

(Report Number)
甲第8641号

(URL)
https://hdl.handle.net/20.500.14094/0100482389

Doctoral Dissertation

Efficient Learning of Neural Networks for Image Processing

画像処理の高度化に向けた ニューラルネットワークの効率的学習法

January 2023

Graduate School of Engineering, Kobe University

Kazuya Urazoe

Abstract

This dissertation concerns the efficient learning of neural networks for image processing.

To improve neural network-based image processing, three important factors must be considered: the (i) image dataset, (ii) network architecture, and (iii) learning algorithm. Factors (i) and (iii) affect the accuracy and robustness of applications, whereas (ii) affects the accuracy and processing speed of applications.

This study focuses on the above three factors in the following applications:

- image super-resolution (in Chapters 2, 3, 4)
- image recognition (in Chapters 5, 6)

Image super-resolution is a resolution conversion method that upsamples a low-resolution image into high resolution. This technique is essential for cameras, televisions, computer monitors, and smart phones.

Image recognition is a technique that extracts features from images and predicts their categories. This technique is essential for optical character recognition, monitoring systems, robot vision, and autonomous driving.

For image super-resolution, this study proposes the following efficient learning methods:

- Chapter 2: (ii) network architecture
- Chapter 3: (iii) learning algorithm
- Chapter 4: (ii) network architecture and (iii) learning algorithm

Chapter 2 proposes an efficient network architecture for image quality and reduces processing time. Chapter 3 proposes an efficient learning algorithm to improve luminance isotropy. Chapter 4 proposes both an efficient network architecture and learning algorithm for the multiple image categories while achieving robustness.

For image recognition, this study proposes the following efficient learning methods:

- Chapter 5: (i) image dataset
- Chapter 6: (ii) network architecture and (iii) learning algorithm

Chapter 5 proposes an efficient generation method on an image dataset and pixel-wise image recognition. Chapter 6 proposes both an efficient network architecture and learning algorithm for rotational-invariant image recognition.

From these, the performance of neural network-based image processing is drastically improved in this study.

Contents

List of Figures

List of Tables

1 Introduction

In the last decade, neural network-based image processing has achieved state-of-the-art performances. In particular, convolutional neural networks (CNNs) are widely used in tasks such as in classification, object detection, semantic segmentation, super-resolution, image generation.

In image recognition, Krizhevsky et al. achieved the top score in the ImageNet Large Scale Visual Recognition Challenge (ILSVRSC) using a CNN in 2012 [1]. Subsequently, many image processing techniques with CNNs have been proposed.

In super-resolution, Dong et al. proposed a super-resolution using CNN (SRCNN) in 2014 [2, 3]. A trained CNN can accurately estimate the detailed information of an image.

However, neural network-based image processing has several unknown parts; this is called the "black box" issue. To improve neural network-based image processing, the following three important factors must be considered:

(i) image dataset

(ii) network architecture

(iii) learning algorithm

The image dataset and learning algorithm affect the accuracy and robustness of applications, whereas the network architecture affects the accuracy and processing speed of applications. Thus, this study focuses on the above three factors for the efficient learning. The target applications are as follows:

- image super-resolution (in Chapters 2, 3, 4)
- image recognition (in Chapters 5, 6)

Conventional image super-resolution suffers from the following problems:

- high computational cost for practical devices

- low isotropy for luminance
- low robustness for multiple image categories

First, conventional super-resolution requires a large computational cost and is difficult to run on practical devices such as televisions and smart phones. Second, they have minimal luminance isotropy and degrade the image quality. Finally, they are difficult to adapt for multiple image categories, such as nature, manga, and text images, using a single neural network.

This dissertation proposes the following methods to solve these issues:

- multi-path network architecture for an efficient computational cost (in Chapter 2)
- luminance inversion averaging method for improving isotropy (in Chapter 3)
- multi-task learning for multiple image categories (in Chapter 4)

Next, conventional image recognition suffers from the following problems:

- difficulty in preparing an image dataset for pixel-wise image recognition
- lack of rotational invariance

First, pixel-wise image recognition requires pixel-based training images and teaching signals. Moreover, collecting various training samples and manually annotating teaching signals incurs significant costs. Second, neural network-based image recognition methods are not invariant to a large rotation transformation on the input image without supports.

This dissertation proposes the following methods to solve these issues:

- synthetic image generation for pixel-wise image recognition (in Chapter 5)
- learning algorithm for improving rotational invariance of image recognitions (in Chapter 6)

Figure 1.1 shows the contents of this dissertation.

The remainder of this dissertation is organized as follows. Chapter 2 proposes an image super-resolution with a CNN using multiple paths. In recent decades, neural network-based super-resolution tends to increase its depth and parameters. However, low computational cost and high-speed processing are essential for practical super-resolution in an environment without an accelerator such as a graphics processing unit (GPU). Therefore, a CNN architecture with

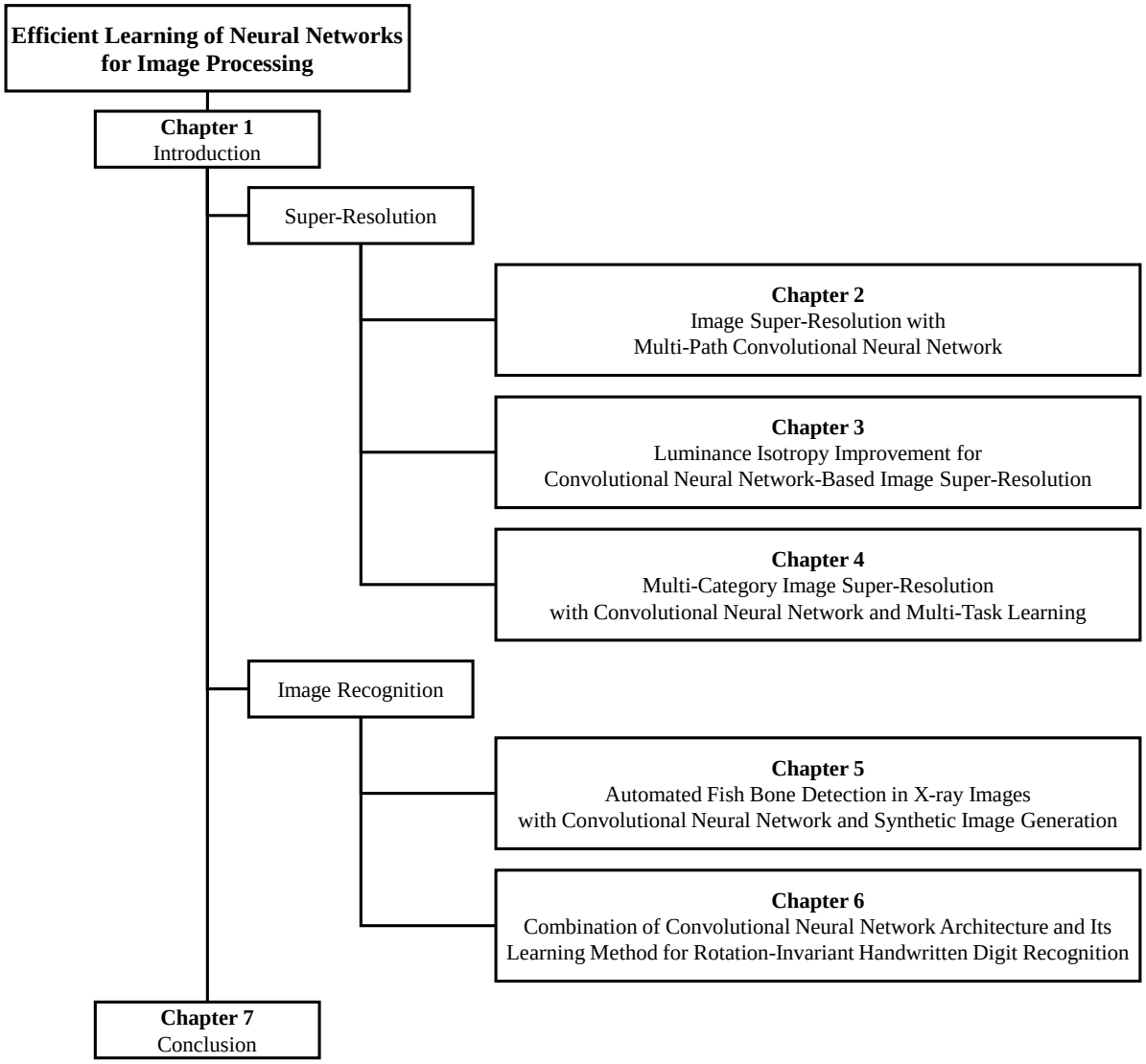


Figure 1.1: Contents of this dissertation.

less parameters and a lower computation is required. The proposed method designs a CNN architecture with multiple paths from the low to deep layer. This architecture can generate and combine components from low to high frequencies independently. Consequently, although the number of propagation paths increases, the number of parameters relative to the performance can be reduced, and the computational cost of super-resolution processing can be reduced without image degradation.

Chapter 3 proposes a method that improves the luminance isotropy of CNN-based image super-resolution. CNN-based image super-resolution is widely used as a high-quality image-

enhancement technique. However, in general, it demonstrates minimal luminance isotropy. Therefore, this chapter proposes two methods, "luminance inversion training" and "luminance inversion averaging," to improve the luminance isotropy of CNN-based image super-resolutions.

Chapter 4 proposes a multi-category image super-resolution using CNN and multi-task learning. The image categories include nature, manga, and text images. Their features differ from each other. However, several CNNs for super-resolution are trained with a single category. If the input image category differs from that of the training images, the performance of super-resolution degrades. Two possible solutions are available for managing multiple categories with conventional CNNs. The first involves preparing CNNs for every category. However, this requires a category classifier to select an appropriate CNN. The second solution involves learning all categories with a single CNN. In this solution, the CNN cannot optimize its internal behavior for each category. Therefore, this chapter presents a super-resolution CNN architecture for multiple image categories. The proposed CNN has two parallel outputs for a high-resolution image and a category label. The main CNN for the high-resolution image has typical architecture with three convolution layers, and the sub-neural network for the category label branches from its middle layer and consists of two fully connected layers. This architecture can simultaneously learn the high-resolution image and its category using multi-task learning. The category information is used to optimize the super-resolution. In an application setting, the proposed CNN can automatically estimate the input image category and change the internal behavior.

Chapter 5 proposes a new fish bone detection technique using a CNN and synthetic image generation. Semantic segmentation CNNs with supervised learning generally require a large number of training images and their pixel-wise teaching signals. Two problems arise in fish bone detection when using semantic segmentation CNNs. One is the manual annotations of fish bones, and the other is the difficulty of sampling all variations of fish bones with various lengths, angles, and thicknesses. Therefore, the proposed method generates these by drawing virtual fish bones on X-ray images. This technique is useful for reducing the cost of collecting and annotating a dataset.

Chapter 6 presents several combinations of a CNN and learning methods for rotation-invariant digit recognition. Rotation data augmentation is widely used for improving rotation invariance. Data augmentation commonly assigns the same label to all augmented images of the same source. However, this learning method causes some collisions between original and rotated digits. Therefore, this chapter presents three types of rotation-invariance learning methods and

applies them to five popular CNN architectures.

Chapter 7 concludes this study and discusses future work.

2 Image Super-Resolution with Multi-Path Convolutional Neural Network

2.1 Introduction

Since 4K/8K broadcasting began in Japan in 2018, the resolutions of various display devices such as televisions and smart phones become increasingly higher. However, when low-resolution images are input to high-resolution display devices, some resolution conversion is required.

Conventional interpolation methods have problems with image degradation such as blurring and artifacting when the magnification factor is high. In general, according to the sampling theorem, signals that exceed the Nyquist frequency (hereinafter referred to as high-frequency signals) are excluded in the original images. Although Bicubic interpolation can generate signals below the Nyquist frequency (hereinafter referred to as low-frequency signals), it does not output high-frequency signals in principle.

Freeman et al. proposed a method that adds high-frequency to low-frequency image upsampled using the interpolation method [4]. This method is based on a database of the high-frequency signals corresponding to the low-frequency signals and is maintained as a dictionary. This is called an example-based super-resolution. However, the size of the dictionary tends to be larger for improving the image quality. Several methods have been proposed to reduce the dictionary size and search time [5–8]. Perez-Pellitero et al. proposed patch symmetry collapse (PSyCo) [7]. They focus on the symmetry of the patches in the dictionary and reduces horizontal, vertical, rotational, and other patch variables to improve the efficiency of dictionary search.

Moreover, Dong et al. proposed a super resolution method using a CNN (SRCNN) [2, 3]. A trained CNN can estimate the detailed information of an image with high accuracy. Following the proposal of the SRCNN, various CNN-based super-resolution methods have been proposed [2, 3, 9–17]. Kim et al. improved image quality by learning the difference from pre-interpolated image to a high-resolution image using a 20-layer deep CNN [10]. Dong et al. introduced a deconvolution layer in the final layer of the CNN, rather than pre-interpolation [11]. By contrast, efficient sub-pixel CNN (ESPCN) [12] proposed by Shi et al. and multi-channel CNN

(MCH) [13, 14] proposed by Ohtani et al. showed architectures in which the final layer of the CNN is multi-output. $K\times$ image magnification can be achieved by a structure that outputs K^2 pixels per input pixel. MCH generates K^2 images of the same size as the input image using a single CNN and arranges them. These methods without temporary enlargement can save memory and process faster because the area of the feature map is small.

However, CNN-based super-resolution typically has a processing time problem. Reference [15] explains the relationship between computational cost and image quality for various types of super-resolution and shows that CNN-based super-resolution is still less efficient than example-based super-resolution and linear interpolation. Furthermore, Reference [16] shows that, in recent CNN-based super-resolution, the image quality improves as the depth increases; however, the computational cost increases exponentially. SRCNN has approximately 57,000 internal parameters, whereas VDSR [10] and RDN [17] have approximately 665,000 and 22.6 million, respectively

Low computational costs and high processing speeds are essential for practical super-resolution in an environment without accelerators such as GPUs. Therefore, a CNN architecture that has less parameters without degrading image quality is required. Recently, CNN-based super-resolution has tended to be multi-layered. Most computations are used for estimating high-frequency signals. The network architecture of CNNs is too sophisticated for generating only low-frequency signals. However, many CNNs are designed to generate both low- and high-frequency signals in a single network. Investigating a CNN architecture with multiple independent paths from low to deep layers is important. This technique increases the number of propagation paths; however, reducing the number of parameters relative to the image quality and computational cost of the super-resolution processing seems possible.

Section 2.2 provides an overview of the conventional method of super-resolution using multi-output CNN (MCH) and its problems. Section 2.3 proposes two super-resolution methods (2Path-CNN and 3Path-CNN) using CNNs with multiple propagation paths. Section 2.4 evaluates the usefulness of the proposed method and compares it with various super-resolution methods.

2.2 Conventional Method

This section overviews the MCH model, proposed by Ohtani et al. [13, 14].

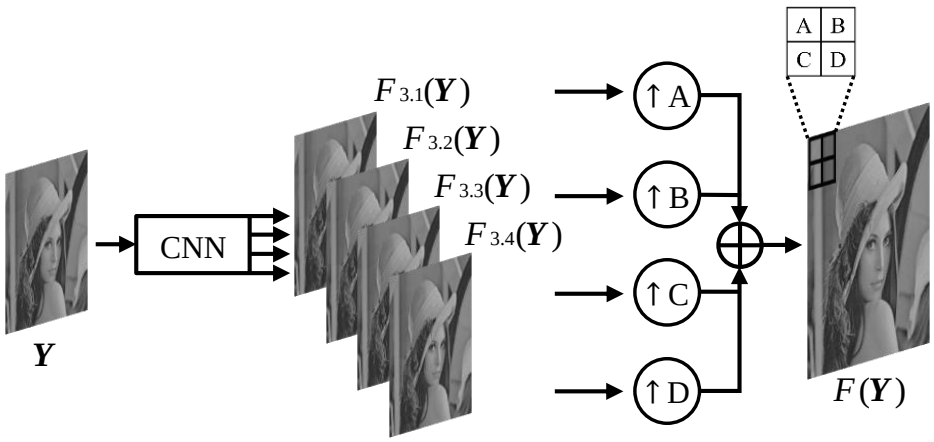


Figure 2.1: Multi-channel CNN. $\boldsymbol{Y}$ and $F(\boldsymbol{Y})$ represent the low-resolution and super-resolution images, respectively. $F_{3.1}(\boldsymbol{Y})$, $F_{3.2}(\boldsymbol{Y})$, $F_{3.3}(\boldsymbol{Y})$, and $F_{3.4}(\boldsymbol{Y})$ denote separate channels of $F_3(\boldsymbol{Y})$. ↑ A, ↑ B, ↑ C, and ↑ D indicate upsampling to their locations on the magnified image, as shown in Figure 2.4

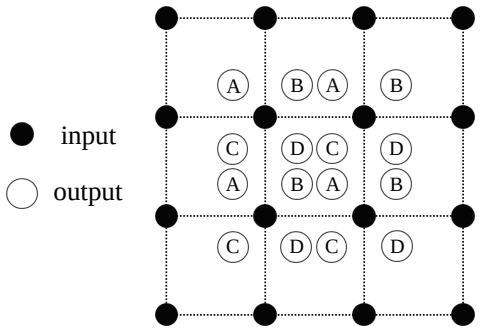


Figure 2.2: Input/output pixel location (2× magnification).

2.2.1 Multi-Channel Convolutional Neural Network for Image Super-Resolution

Figure 2.1 presents an overview of the MCH. Let K be the magnification ratio. To obtain a $K\times$ magnified image, MCH generates $K \times K$ pixels per input pixel. For $2\times$ magnification, there are four types of output pixel locations, as shown in Figure 2.2: (A) upper left, (B) upper right, (C) lower left, and (D) lower right of the nearest input pixel. These output pixels are generated from K^2 output channels of the single CNN. Figure 2.3 shows the architecture of MCH.

Let $\boldsymbol{Y}$ be the low-resolution image. The output of the first convolution layer are calculated

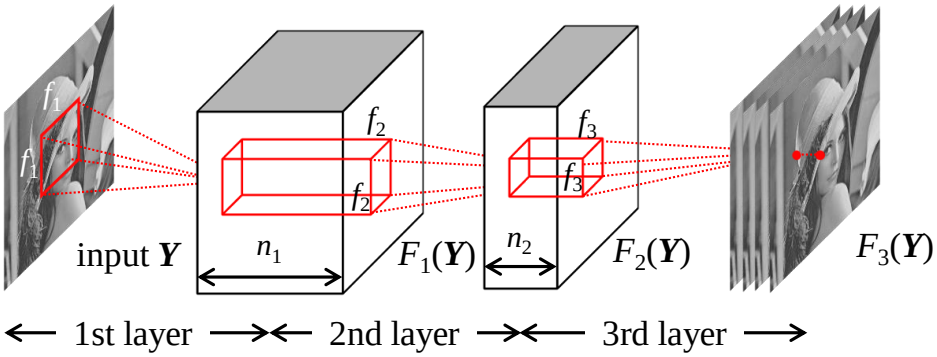


Figure 2.3: Architecture of MCH.

as follows:

$$F_1(\boldsymbol{Y}) = A(W_1 * \boldsymbol{Y} + B_1), \tag{2.1}$$

where W_1 is the convolution filter, B_1 the bias, A the activation function, and "$*$" denotes the convolution operation. W_1 is a four-dimensional tensor, and its size is $1 \times f_1 \times f_1 \times n_1$, where $f_1 \times f_1$ is the filter size, and n_1 is the number of filters in the first convolution layer. In MCH, activation function A is the rectified linear unit (ReLU) [18]. The output of the second convolution layer are calculated as follows:

$$F_2(\boldsymbol{Y}) = A(W_2 * F_1(\boldsymbol{Y}) + B_2). \tag{2.2}$$

The size of W_2 is $n_1 \times f_2 \times f_2 \times n_2$, where $f_2 \times f_2$ is the filter size and n_2 is the number of filters in the second convolution layer. Finally, the K^2 pixels are calculated as

$$F_3(\boldsymbol{Y}) = W_3 * F_2(\boldsymbol{Y}) + B_3, \tag{2.3}$$

where $F_3(\boldsymbol{Y})$ has K^2 channels. The size of W_3 is $n_2 \times f_3 \times f_3 \times K^2$. Let $F_{3.1}(\boldsymbol{Y})$, $F_{3.2}(\boldsymbol{Y})$, $\cdots$, $F_{3.K^2}(\boldsymbol{Y})$ be separate channels of $F_3(\boldsymbol{Y})$. Then, the super-resolution image, $F(\boldsymbol{Y})$, can be obtained using

$$F(\boldsymbol{Y}) \quad = \quad \uparrow \mathrm{A}(F_{3.1}(\boldsymbol{Y})) + \uparrow \mathrm{B}(F_{3.2}(\boldsymbol{Y})) + \cdots, \tag{2.4}$$

where $\uparrow \mathrm{A}(\ ), \uparrow \mathrm{B}(\ ), \cdots$ denote upsampling to their locations on the magnified image. Figure 2.4 shows an example of a 2×2 synthesis. In MCH, none of the convolution layers have padding to prevent image border effects.

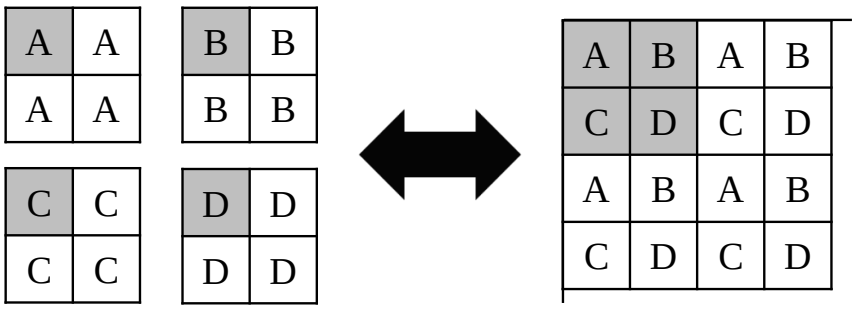


Figure 2.4: Separation and synthesis of pixels (2× magnification).

2.2.2 Training Method

In the training phase, the original high-resolution images and their images down-sampled by Bicubic interpolation are prepared. The CNN is trained with low-resolution image $\boldsymbol{Y}$ as the input signal and high-resolution image $\boldsymbol{T}$ as the teaching signal. The loss function of MCH is the mean squared error (MSE) given by

$$L_{\mathrm{MSE}}(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|F(\boldsymbol{Y}_i;\Theta) - \boldsymbol{T}_i\|^2, \tag{2.5}$$

where N is the number of training data samples and Θ denotes the parameters to be determined using a backpropagation technique such as W_1, W_2, W_3, B_1, B_2, and B_3. The learning rate is 10^{-4} for the first and second convolution layers, and 10^{-5} for the third convolution layer, as in Reference [14].

2.2.3 Problems

Theoretically, CNNs have a calculation ability equivalent to that of Bicubic interpolation with a single convolution layer. Indeed, as the experiment in Section 2.4.1 proves, a single-layer CNN can achieve a slightly higher peak signal to noise ratio (PSNR) than that of Bicubic interpolation. This suggests that a two-layer CNN over-processes when generating only low-frequency signals. The deep CNNs is effective for estimating high-frequency signals. Nevertheless, many conventional methods generate low- and high-frequency signals simultaneously in a single path. If the low-frequency signals are generated by a low-layer of a CNN and the high-frequency signals are generated by a deep-layer of the CNN, the number of parameters relative to the image quality

could be reduced. This idea helps reduce processing time while maintaining image quality.

2.3 Image Super-Resolution with Multi-Path Convolutional Neural Network

This section proposes a CNN architecture with multiple propagation paths from low to deep layers and independently generates low-frequency signals to high-frequency signals. Furthermore, this section proposes a method for learning each propagation path sequentially. The cutoff frequencies of low, middle, and high do not clearly appear in the proposed CNN architecture; this chapter refers to the low-layer CNN as the path that generates low-frequency signals, and the deep-layer CNN as the path that generates high-frequency signals. Section 2.3.1 shows the five types of CNN structures, and Section 2.3.2 explains how to learn them.

2.3.1 Multi-Path Convolutional Neural Networks

This section explains the five types of CNN structures:

- 1Layer-CNN
- nLayer-CNN
- 2Layer2Path-CNN
- 3Layer2Path-CNN
- 3Path-CNN

1Layer-CNN

Figure 2.5 shows a 1Layer-CNN. For low-resolution image $\boldsymbol{Y}$, the outputs of the K^2 channels is calculated as

$$F_1(\boldsymbol{Y}) = W_1 * \boldsymbol{Y} + B_1. \tag{2.6}$$

It consists of one convolution layer and has no feature maps in the intermediate layers. In addition, its 4×4 filter has a calculation ability almost equivalent to that of Bicubic interpolation. As the experiment in Section 2.4.1 proves, the 1Layer-CNN can achieve a slightly higher PSNR than that of Bicubic interpolation.

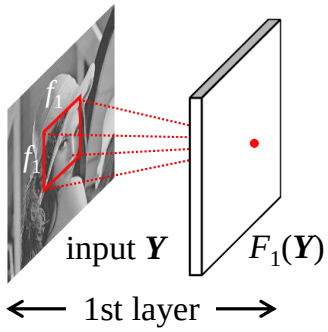


Figure 2.5: 1Layer-CNN.

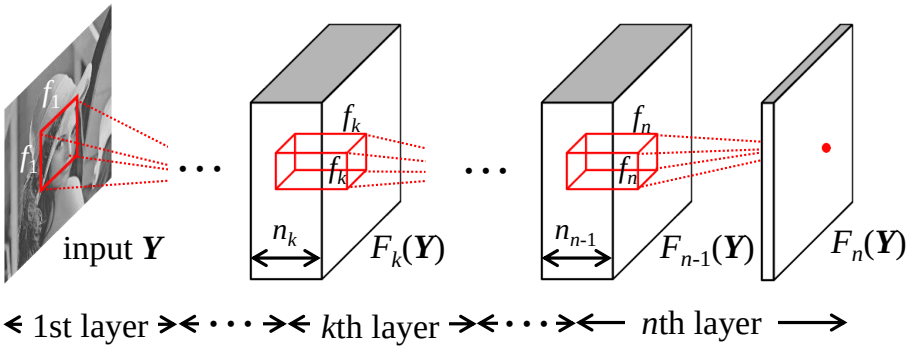


Figure 2.6: nLayer-CNN.

nLayer-CNN

This section explains a generalization of the MCH in Section 2.1, comprising n convolution layers. Figure 2.6 shows the nLayer-CNN.

For low-resolution image $\boldsymbol{Y}$, the kth $(1 \leq k \leq n-1)$ feature maps $F(\boldsymbol{Y}_k)$ are calculated as follows:

$$F_k(\boldsymbol{Y}) = A(W_k * F_{k-1}(\boldsymbol{Y}) + B_k). \tag{2.7}$$

The size of W_k is $n_{k-1} \times f_k \times f_k \times n_k$, where $f_k \times f_k$ is the filter size and n_k is the number of filters in the kth convolution layer. The output of the nth layer is calculated as follows:

$$F_n(\boldsymbol{Y}) = W_n * F_{n-1}(\boldsymbol{Y}) + B_n. \tag{2.8}$$

A CNN can extract more complicated features with deeper convolution layers. Therefore, to estimate more complicated high-frequency signals, increasing the number of convolution layers

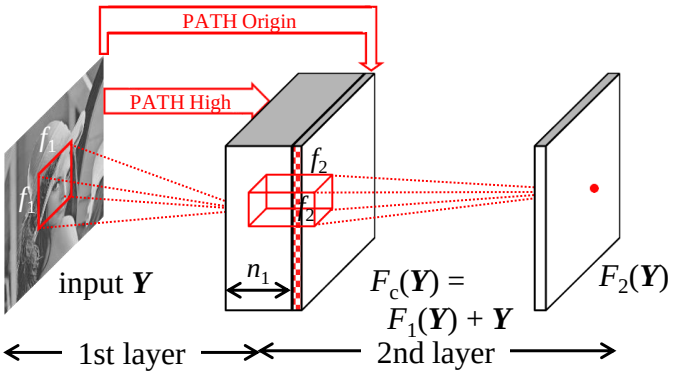


Figure 2.7: 2Layer2Path-CNN.

is a reasonable strategy. However, as the experiment in Section 2.4.1 proves, the image quality for $n > 3$ is degraded compared with that obtained using a 3Layer-CNN. This is because the gradient loss problem caused as the CNN becomes deeper. In addition, to generate low-frequency signals, a CNN with more than two layers is excessive processing. Deep CNNs are effective for estimating high-frequency signals. Therefore, for fast super-resolution processing, this chapter focuses on the frequency bands and the architecture that shallow and deep CNNs to specialize in estimating low- and high-frequency signals, respectively.

2Layer2Path-CNN

Figure 2.7 shows a 2Layer2Path-CNN. It is based on the 2Layer-CNN and consists of two paths: *PATH High*, which propagates high-frequency signals, and *PATH Origin*, which propagates input image $\boldsymbol{Y}$.

PATH Origin propagates input image $\boldsymbol{Y}$, while *PATH High* propagates high-frequency signals. The first-layer feature maps $F_1(\boldsymbol{Y})$ that have high-frequency signals is obtained by *PATH High* using Equation 2.1 in MCH. Then, the feature map $F_1(\boldsymbol{Y})$ propagated from *PATH High* and input image $\boldsymbol{Y}$ propagated from *PATH Origin* are combined on the feature axis as follows:

$$F_c(\boldsymbol{Y}) = F_1(\boldsymbol{Y}) + \boldsymbol{Y}. \tag{2.9}$$

Finally, the K^2 pixels are calculated as follows:

$$F_2(\boldsymbol{Y}) = W_2 * F_c(\boldsymbol{Y}) + B_2. \tag{2.10}$$

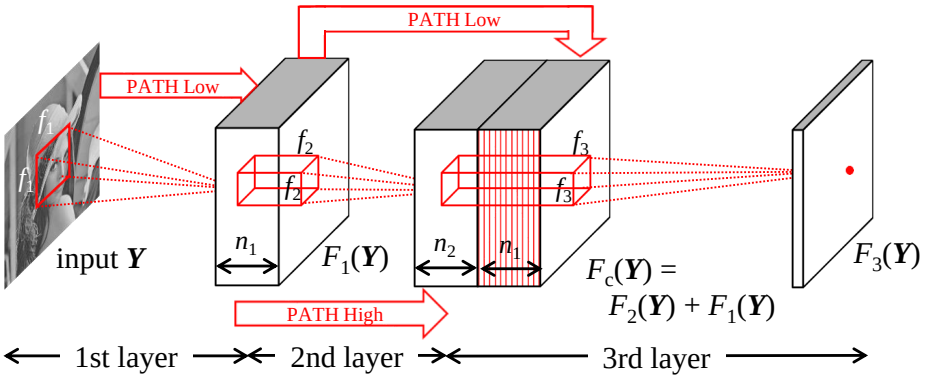


Figure 2.8: 3Layer2Path-CNN.

3Layer2Path-CNN

Figure 2.8 shows a 3Layer2Path-CNN. It is based on the 3Layer-CNN and consists of two paths: *PATH Low*, which propagates low-frequency signals and *PATH High*, which propagates high-frequency signals.

PATH Low generates feature maps $F_1(\boldsymbol{Y})$ that have low-frequency signals using Equation 2.1 for MCH. *PATH High* generates feature maps using $F_1(\boldsymbol{Y})$ from *PATH Low* with Equation 2.2 for MCH. *PATH High* generates feature maps $F_2(\boldsymbol{Y})$ that have high frequency signals. The sets of feature maps $F_1(\boldsymbol{Y})$ and $F_2(\boldsymbol{Y})$ propagated from *PATH Low* and *PATH High* are combined on the feature axis, and the feature maps with both low- and high frequency signals are generated as

$$F_c(\boldsymbol{Y}) = F_2(\boldsymbol{Y}) + F_1(\boldsymbol{Y}). \tag{2.11}$$

Finally, the K^2 pixels are calculated as

$$F_3(\boldsymbol{Y}) = W_3 * F_c(\boldsymbol{Y}) + B_3. \tag{2.12}$$

3Path-CNN

Figure 2.9 shows a 3Path-CNN. It is based on the 3Layer-CNN and consists of three paths: *PATH Low*, which propagates low-frequency signals; *PATH High*, which propagates high-frequency signals; and *PATH Origin*, which propagates the input image $\boldsymbol{Y}$.

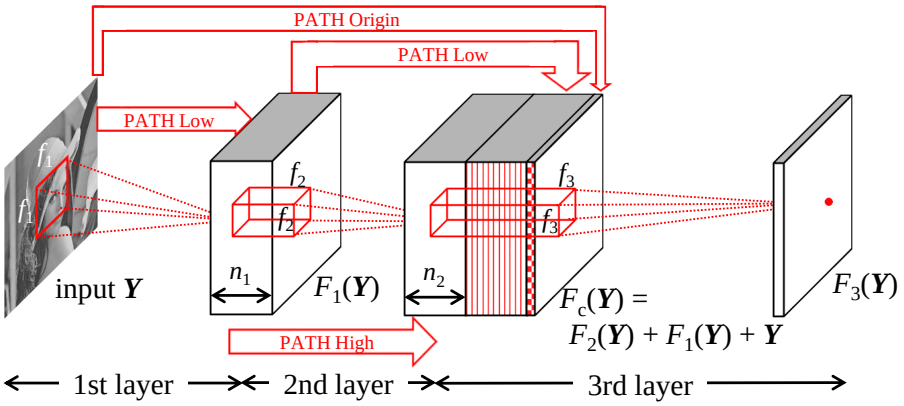


Figure 2.9: 3Path-CNN.

PATH Low generates first feature maps $F_1(\boldsymbol{Y})$ that have low-frequency signals, using Equation 2.1 in MCH. *PATH High* generates feature maps $F_2(\boldsymbol{Y})$ that includes high-frequency signals using $F_1(\boldsymbol{Y})$ from *PATH Low* and Equation 2.2. Then, feature maps $F_1(\boldsymbol{Y})$ from *PATH Low*, feature maps $F_2(\boldsymbol{Y})$ propagated from *PATH High*, and input image $\boldsymbol{Y}$ are combined on the feature axis as

$$F_c(\boldsymbol{Y}) = F_2(\boldsymbol{Y}) + F_1(\boldsymbol{Y}) + \boldsymbol{Y}. \tag{2.13}$$

Finally, the K^2 pixels are calculated as

$$F_3(\boldsymbol{Y}) = W_3 * F_c(\boldsymbol{Y}) + B_3. \tag{2.14}$$

These five methods introduce the Parametric ReLU (PReLU) [19] as activation function A and batch normalization [20] after the second layer of a CNN with more than three layers.

The use of multiple propagation paths has two purposes. First, convolution on *PATH High* is specialized for generating high-frequency signals. The second and later layers do not need to generate low-frequency signals. Consequently, the number of feature maps can be reduced. Second, input image $\boldsymbol{Y}$ is propagated on *PATH Origin*, and the low-frequency signals equivalent to Bicubic interpolation can be propagated over the shortest path. From this, *PATH Origin* helps improve the image quality.

Note that the proposed method does not use residual learning [21]. This technique is widely used in recent CNN-based super-resolution [10, 16, 17]. VDSR [10] learns the difference between pre-interpolated images and high-resolution images using residual learning. However, residual learning is equivalent to propagating the pre-enlarged image with a path whose weight is fixed

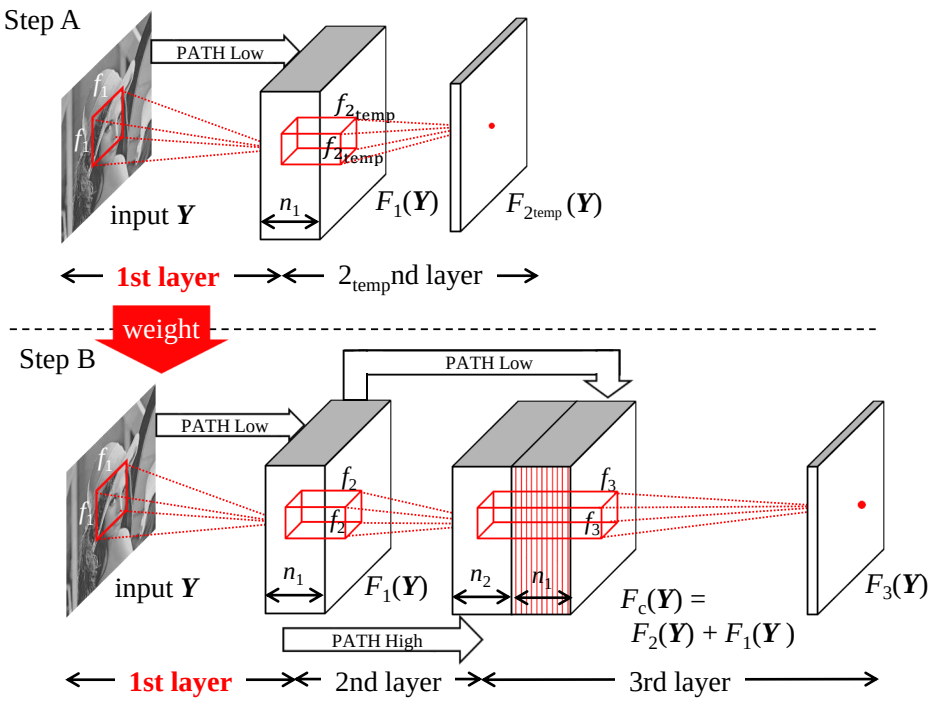


Figure 2.10: Step-learning for the 2Path-CNN.

at 1. That is, the coefficients are not updated for this path and out of learning process range. Conversely, in the proposed method, all propagation paths have filters and within the range of the learning process. This is the difference between the proposed methods and residual learning-based super-resolutions.

2.3.2 Step Learning for Frequency Band Separation

In the 2Path-CNN and 3Path-CNN, the CNN has multiple propagation paths from low to deep layers and generates low- and high-frequency signals independently. However, the propagation paths naturally separate into different frequency bands with difficulty because the parameters of CNN are initialized with random values. For this issue, the 3Path-CNN and 2Path-CNN pre train the 2Path-CNN and 1Path-CNN in advance, respectively.

Figures 2.10 and 2.11 show an overview of the two-step learning in the 2Path-CNN and 3Path-CNN, respectively. In this method, the CNN is trained in two steps: A and B.

For example, as shown in Figure 2.10, the 2Path-CNN pre trains 2Layer-CNN in *Step A*. Next, in *Step B*, the pre trained filter W_1 and bias B_1 of *Step A* are used for 2Path-CNN. The

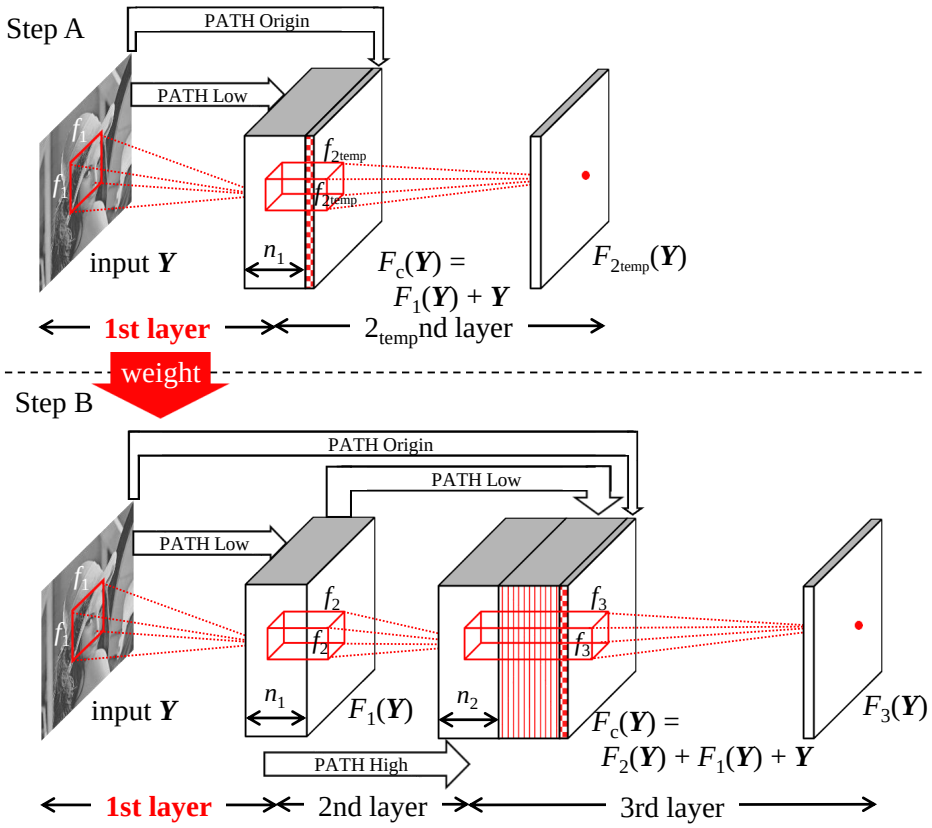


Figure 2.11: Step-learning for the 3Path-CNN.

learning rate of the first layer is set to half that of the second layer, and the entire CNN is trained while maintaing the pre trained status of the first layer. Consequently, *PATH Low* and *High* are specialized for the low- and high-frequency, respectively.

2.4 Evaluation Experiments

In this section, we evaluate the usefulness of the proposed methods: 2Path-CNN and 3Path-CNN. Section 2.4.1 verifies the performance limitations for the single-path CNN. Section 2.4.2 evaluates the architectures that generates low- to high-frequency components independently using CNNs with multiple paths. Section 2.4.3 examines the frequency response of a circular zone plate image. Section 2.4.4 evaluates the relationship between image quality and processing time when combining the number of feature maps in the 3Path-CNN. Section 2.4.5 compares the performance of eight super-resolution techniques.

The experimental environment is as follows—Operating system (OS): Ubuntu 16.04 LTS, Central processing unit (CPU): Intel Core i7-8700 3.20GHz, Memory: 16.00GB, GPU: NVIDIA GeForce GTX 1080 8GB, and Integrated development environment (IDE): MATLAB.

All CNNs are implemented with a Caffe package [22]. All experiments focus only on the luminance channel in the YCrCb space because several studies on super-resolution have evaluated the luminance channel alone [3, 10, 11, 13, 14, 23–27]. SRCNN [3] and FSRCNN [11], which are compared in Section 2.4.5, are also implemented in Caffe [22].

2.4.1 Performance Limitation of the nLayer-CNN with a Single Path

This section shows that the 1Layer-CNN can achieve the same PSNR or better than Bicubic interpolation even with the performance limitation of single-path CNN. The performance limitation of CNNs that learn the difference between pre-enlarged and high-resolution images have already been reported in Reference [3]. This experiment verifies the performance of CNNs that magnify images at the final stage.

The five evaluation methods were as follows:

- Bicubic interpolation
- 1Layer-CNN
- 2Layer-CNN
- 3Layer-CNN
- 4Layer-CNN

The filter size $f_k \times f_k$ of the CNN was set to the middle layer $f_k = 5$ and output layer $f_n = 3$, as in Reference [14]. Table 2.1 shows the feature maps for each method. In the 2Layer-CNN, the number of feature maps in the first layer changed as $n_1 = 8, 16, 32$, and 64. Then, this section evaluates the relationship between image quality and processing time. In the 3Layer-CNN and 4Layer-CNN, the number of feature maps in the middle layer was set to $n_k = 16$. Hereinafter, the CNN with the number of feature maps $n_1 = 16$, $n_2 = 16$, and $n_3 = 4$ is denoted as (16-16-4). In this experiment, the CNN is trained on the T91 [28] dataset, which is a dataset of natural images. The input signal, which is a low-resolution image, is downsampled to $1/K$ using Bicubic interpolation according to magnification factor K. The number of backpropagations is

Table 2.1: Evaluation methods for CNNs with a single path.

Method	Number of feature maps		
	n_1	n_2	n_3
1Layer-CNN (4)	-	-	-
2Layer-CNN (8-4)	8	-	-
2Layer-CNN (16-4)	16	-	-
2Layer-CNN (32-4)	32	-	-
2Layer-CNN (64-4)	64	-	-
3Layer-CNN (16-16-4)	16	16	-
4Layer-CNN (16-16-16-4)	16	16	16

10 million at its maximum. The learning rate is 10^{-4} for all layers, except for the final layer, which is 10^{-5}. In addition, the learning rate for each layer was set to three-fourths after 7.5 million backpropagations. The image quality is evaluated with the BSDS100 [29] dataset, which is a dataset of natural images.

The evaluation targets are image quality and processing time. Image quality is objectively evaluated with PSNR and structural similarity (SSIM), which compare the original and super-resolution images.

Let $\boldsymbol{Y}_{\mathrm{HR}}$ and $\boldsymbol{Y}_{\mathrm{SR}}$ be the 8-bit high- and super-resolution images, respectively. The PSNR evaluates the peak ratio between two images and is calculated as follows:

$$PSNR = 10log_{10}\frac{255^2}{\frac{1}{xy}\sum_{i=1}^{x}\sum_{j=1}^{y}(\boldsymbol{Y}_{\mathrm{HR},ij} - \boldsymbol{Y}_{\mathrm{SR},ij})^2}, \tag{2.15}$$

where x and y are the number of columns and rows in the image, respectively, and $\boldsymbol{Y}_{ij}$ denotes the pixel value of the image at index (i, j)

The SSIM evaluates the similarity of structures between two images and is calculated as follows:

$$SSIM = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{2.16}$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} denote the local average of $\boldsymbol{Y}_{\mathrm{HR}}$, local average of $\boldsymbol{Y}_{\mathrm{SR}}$, standard deviation of $\boldsymbol{Y}_{\mathrm{HR}}$, standard deviation of $\boldsymbol{Y}_{\mathrm{SR}}$, and mutual covariances of $\boldsymbol{Y}_{\mathrm{HR}}$ and $\boldsymbol{Y}_{\mathrm{SR}}$,

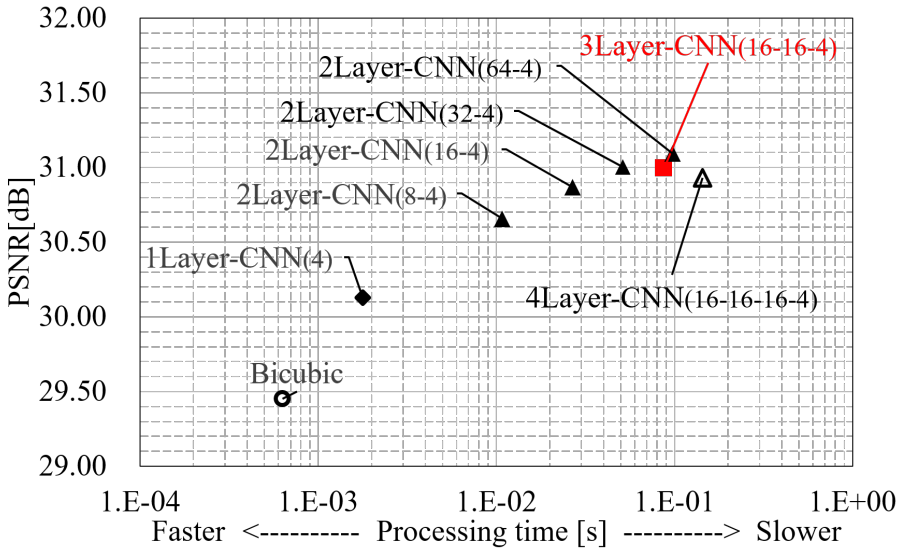


Figure 2.12: PSNRs and processing times for single-path CNNs. The horizontal and vertical axes are the processing time and average PSNR of BSDS100, respectively.

respectively. C_1 and C_2 are nomarization values calculated as follows:

$$C_1 = (0.01 * 255)^2, C_2 = (0.03 * 255)^2. \tag{2.17}$$

Figure 2.12 shows the performance of each CNN for 2× magnification. First, this section focuses on Bicubic interpolation and the 1Layer-CNN. The 1Layer-CNN achieved a PSNR higher than that of Bicubic interpolation by 0.68 dB; however, the processing time increased. Bicubic interpolation enlarges signals below the low-frequency signals in principle. Therefore, the 1Layer-CNN is sufficient for generating only low-frequency signals.

This section focuses on the 2Layer-CNNs. As shown in Figure 2.12, the PSNR improved as the number of feature maps n_1 in the first layer increased. Therefore, increasing the number of feature maps is a strategy for improving image quality. However, the growth rate of the PSNR slowed the processing time.

Finally, this section focuses on the 3Layer-CNN and 4Layer-CNN. Even if the numbers of feature maps in the middle convolution layers are the same, the 4Layer-CNN achieved a PSNR lower than that of the 3Layer-CNN by 0.07 dB. This is because of the gradient loss problem caused by the deep architecture of the CNN. From these results, specializing the second and subsequent convolution layers for generating high-frequency signals is a reasonable method for improving image quality assuming that the first convolution layer is sufficient for generating

Table 2.2: Evaluation methods for CNNs with multiple paths.

Method	Feature maps just before the output layer
2Layer-CNN (16-4)	$F_1(\boldsymbol{Y})$
2Layer2Path-CNN (16+1-4)	$F_1(\boldsymbol{Y}) + \boldsymbol{Y}$
3Layer-CNN (16-16-4)	$F_2(\boldsymbol{Y})$
3Layer2Path-CNN (16-16+16-4)	$F_2(\boldsymbol{Y}) + F_1(\boldsymbol{Y})$
3Path-CNN (16-16+16+1-4)	$F_2(\boldsymbol{Y}) + F_1(\boldsymbol{Y}) + \boldsymbol{Y}$

low-frequency signals.

2.4.2 CNNs with Multiple Propagation Paths

This section evaluates the architecture that independently generates low- to high-frequency signals with multiple paths. The five evaluation methods are as follows:

- 2Layer-CNN (16-4)
- 2Layer2Path-CNN (16+1-4)
- 3Layer-CNN (16-16-4)
- 3Layer2Path-CNN (16-16+16-4)
- 3Path-CNN (16-16+16+1-4)

The filter size, number of feature maps, magnification factor, training images, number of backpropagations, learning rate, testing images, and evaluation methods for the CNNs are the same as in Section 2.4.1. Table 2.2 shows the CNN conditions for each method.

This section explains the notation of the CNN architecture with multiple propagation paths. For example, a basic 3Layer-CNN with $n_1 = 24$, $n_2 = 8$, and $n_3 = 4$ feature maps is denoted as (24-8-4). A 2Path-CNN that combines the first layer output $F_1(\boldsymbol{Y})$ and second layer $F_2(\boldsymbol{Y})$ is denoted as (24-8+24-4). A 3Path-CNN that combines the first layer output $F_1(\boldsymbol{Y})$, second layer $F_2(\boldsymbol{Y})$, and input image $\boldsymbol{Y}$ is denoted as (24-8+24+1-4).

Table 2.3, Table 2.4, and Figure 2.13 show the performance of each CNN at 2× magnification.

This section first focuses on the 2Layer-CNN and 2Layer2Path-CNN.

Table 2.3: PSNRs, SSIMs, and processing times of the 2× magnified image of the 2Layer-CNN with multiple paths (BSDS100).

Method	PSNR [dB] / SSIM / Processing time [s]
2Layer-CNN (16-4)	30.87 / 0.8812 / **0.0269**
2Layer2Path-CNN (16+1-4)	**30.90** / **0.8818** / 0.0286

Bold red font indicates the best in all methods.

Table 2.4: PSNRs, SSIMs, and processing times of the 2× magnified image of the 3Layer-CNN with multiple paths (BSDS100).

Method	PSNR [dB] / SSIM / Processing time [s]
3Layer-CNN (16-16-4)	31.00 / 0.8837 / **0.0874**
3Layer2Path-CNN (16-16+16-4)	31.04 / 0.8841 / 0.1051
3Path-CNN (16-16+16+1-4)	**31.18** / **0.8860** / 0.1056

Bold red font indicates the best in all methods.

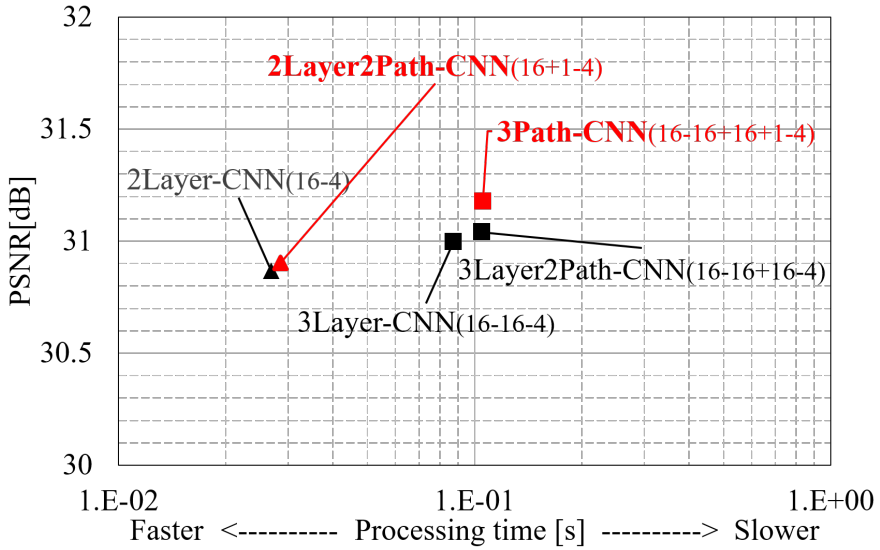


Figure 2.13: PSNRs and processing times for multi-path CNNs. The horizontal and vertical axes are the processing time and average PSNR of BSDS100, respectively.

The 2Layer2Path-CNN achieved a PSNR higher than that of the 2Layer-CNN by 0.03 dB. The feature map $F_c(\boldsymbol{Y})$ just before the final output layer was increased by input image $\boldsymbol{Y}$, and the processing time slightly increased.

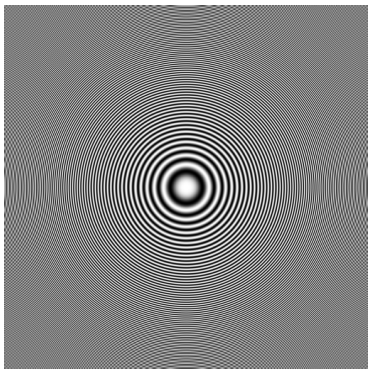


Figure 2.14: Circular zone plate.

This section focuses on the 3Layer-CNN, 3Layer2Path-CNN, and 3Path-CNN. The 3Layer2Path-CNN and 3Path-CNN achieved PSNRs higher than that of 3Layer-CNN by 0.04 dB and 0.18 dB, respectively. Because the low-frequency signals $F_1(\boldsymbol{Y})$ in the first layer and input image $\boldsymbol{Y}$ are propagated, the second convolution layer specialized in generating high-frequency signals. The processing time was slightly increased by the additional $F_c(\boldsymbol{Y})$. In particular, compared with 3Layer2Path-CNN, 3Path-CNN showed a large improvement in PSNR relative to the increase in processing time. This seems to be because the low-frequency signals equivalent to Bicubic interpolation were propagated over the shortest path by *PATH Origin*.

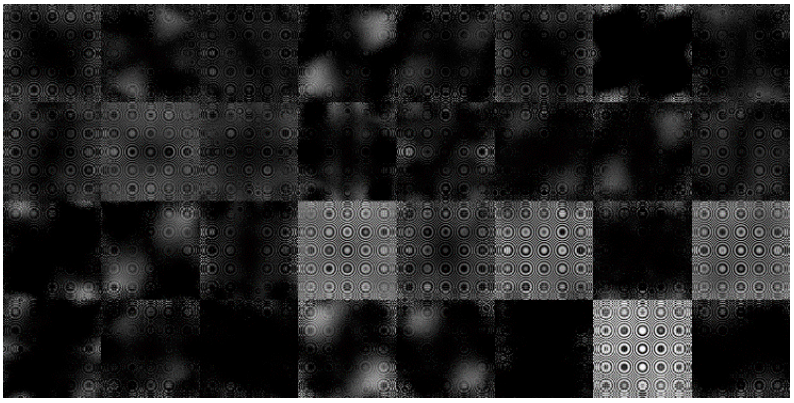
2.4.3 Frequency Response of the Circular Zone Plate Image

This section evaluates the frequency bands of each propagation path. Figure 2.14 shows a circular zone plate (CZP), which is a two-dimensional representation of a sine or cosine curve. In a CZP, the spatial frequencies are increased from the center to the periphery. In this experiment, the CZP image is used as input to the CNN, and the frequency responses are evaluated by visualizing the feature map just before the final output layer.

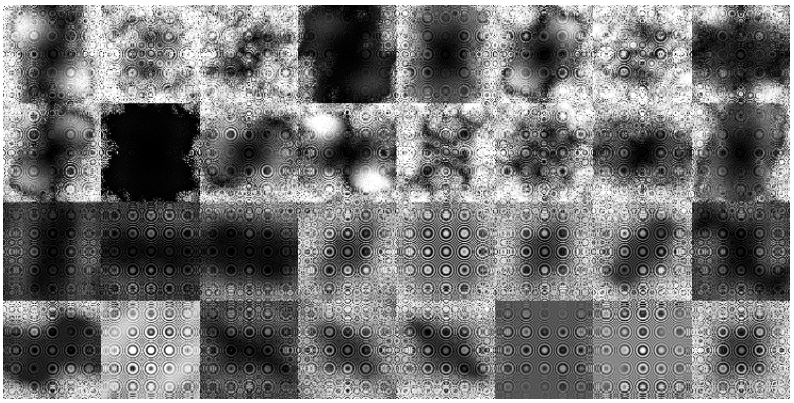
Figure 2.15 shows the following CNN's frequency responses:

- 3Layer-CNN (16-16-4)
- 3Layer2Path-CNN (16-16+16-4)
- 3Path-CNN (16-16+16+1-4)

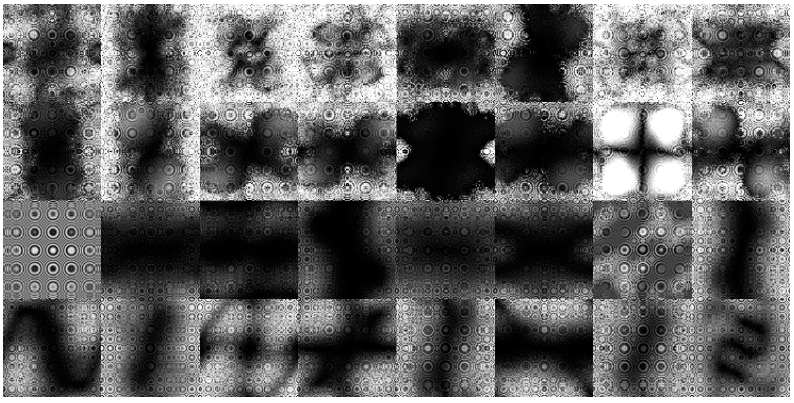
In Figure 2.15-(a), at least, four flat feature maps seem to correspond to low-frequency signals.



(a) 3Layer-CNN (16-16-4).



(b) 3Layer2Path-CNN (16-16+16-4).



(c) 3Path-CNN (16-16+16+1-4).

Figure 2.15: Feature maps just before the final output layer for circular zone plate input.

This shows that the feature maps of the 3Layer-CNN generated both low- and high-frequency signals. In Section 2.4.1, we confirm that a single convolution layer can generate low-frequency signals equivalent to Bicubic interpolation. Therefore, the second and subsequent layers seems to be excessive convolution processing for generating low-frequency signals.

In Figures 2.15-(b) and (c), the 3Layer2Path-CNN and 3Path-CNN propagated the high-frequency signals in the *PATH High* and low-frequency signals in *PATH Low*. Compared to that in Figure 2.15-(a), all feature maps were fully activated, and the feature maps were used effectively.

2.4.4 Relationship between Image Quality and Processing Time in the 3Path-CNN.

In Section 2.4.2, the 3Path-CNN showed good results in balancing between PSNR and processing time. Therefore, this section evaluates the relationship between image quality and processing time by changing the number of feature maps for the 3Path-CNN. The following seven methods had the number of feature maps n_1 and n_2 changed:

- 3Path-CNN (2-30+2+1-4)
- 3Path-CNN (4-28+4+1-4)
- 3Path-CNN (8-24+8+1-4)
- 3Path-CNN (16-16+16+1-4)
- 3Path-CNN (24-8+24+1-4)
- 3Path-CNN (28-4+28+1-4)
- 3Path-CNN (30-2+30+1-4)

Table 2.5 shows the number of feature maps for each method. The filter size, number of feature maps, magnification factor, training images, number of backpropagations, learning rate, testing images, and evaluation methods for the CNNs are the same as in Section 2.4.1.

Table 2.6 and Figure 2.16 show the evaluation results of each method at 2× magnification.

3Path-CNN (8-24+8+1-4) and (16-16+16+16+1-4) showed the highest PSNRs. This indicates that image quality improves when the number of feature maps for the low- and high-frequency

Table 2.5: Evaluation methods for the 3Path-CNN.

Method	Number of feature map	
	n_1	n_2
3Path-CNN (2-30+2+1-4)	2	30
3Path-CNN (4-28+4+1-4)	4	28
3Path-CNN (8-24+8+1-4)	8	24
3Path-CNN (16-16+16+1-4)	16	16
3Path-CNN (24-8+24+1-4)	24	8
3Path-CNN (28-4+28+1-4)	28	4
3Path-CNN (30-2+30+1-4)	30	2

Table 2.6: PSNRs, SSIMs, and processing times of the 2× magnified image with several combinations of feature maps (BSDS100).

Method	PSNR [dB] / SSIM / Processing time [s]
3Path-CNN (2-30+2+1-4)	30.01 / 0.8840 / **0.0560**
3Path-CNN (4-28+4+1-4)	31.15 / 0.8859 / 0.0615
3Path-CNN (8-24+8+1-4)	**31.18** / **0.8863** / 0.0824
3Path-CNN (16-16+16+1-4)	**31.18** / 0.8860 / 0.1056
3Path-CNN (24-8+24+1-4)	31.12 / 0.8854 / 0.1235
3Path-CNN (28-4+28+1-4)	31.06 / 0.8843 / 0.1295
3Path-CNN (30-2+30+1-4)	31.00 / 0.8838 / 0.1315

Bold red font indicates the best in all combinations of feature maps.

signals are designed to be approximately the same. In particular, the 3Path-CNN (8-24+8+1-4) had the shortest processing time. Conversely, 3Path-CNN (4-28+4+1-4) decreased the PSNR by 0.03 dB and had a large reduction in processing time. This also seems to be a good architecture. The following section compares the 3Path-CNN (8-24+8+1-4) and 3Path-CNN (4-28+4+1-4), which showed satisfactory results, as the 3Path-CNN (Type A) and 3Path-CNN (Type B).

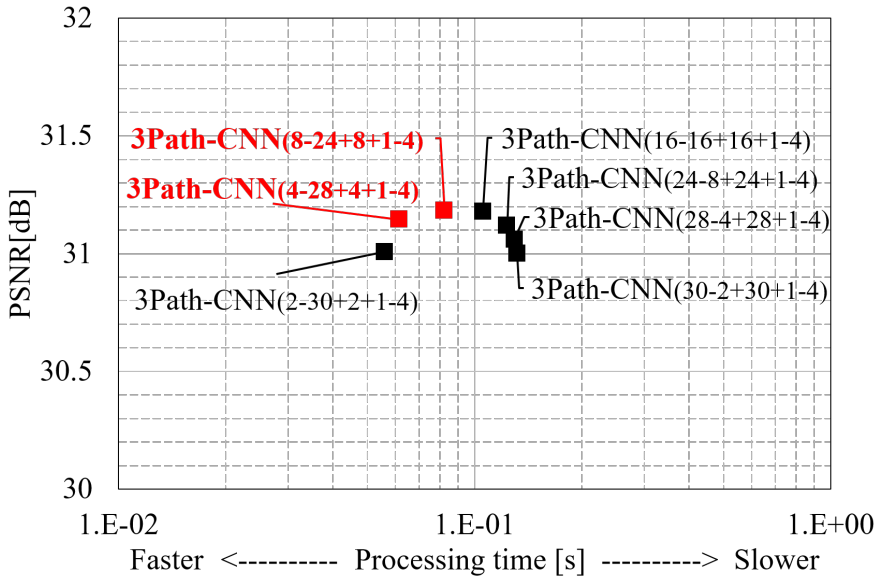


Figure 2.16: PSNRs and processing times for combinations of feature maps (BSDS100, 2× magnification). The horizontal and vertical axes are the processing time and average PSNR, respectively.

2.4.5 Comparison of the Performance for Eight Super-Resolution Techniques

This section compares the conventional method of super-resolution (MCH) and proposed 3Path-CNN. Subsequently, the proposed method is evaluated with seven other high-performance super-resolution processes.

Comparison with the Conventional Method (MCH)

This section compares the 3Path-CNN with the MCH. The four evaluation methods are as follows:

- Bicubic interpolation
- MCH [14]
- 3Path-CNN (Type A)
- 3Path-CNN (Type B)

Table 2.7: Number of feature maps of the CNN for $K\times$ mangification.

Magnification	Number of feature maps $\{n_1, n_2\}$		
	MCH [14]	3Path-CNN (TypeA)	3Path-CNN (TypeB)
2×	{64, 32}	{8, 24}	{4, 28}
3×	{128, 64}	{16, 48}	{8, 56}
4×	{128, 64}	{16, 48}	{8, 56}

Table 2.8: Filter sizes of the CNN for $K\times$ mangification.

Magnification	Filter sizes $\{f_1 \times f_1, f_2 \times f_2, f_3 \times f_3\}$
	MCH [14], 3Path-CNN (TypeA), 3Path-CNN (TypeB)
×2	{5×5, 5×5, 3×3}
×3	{5×5, 3×3, 1×1}
×4	{3×3, 3×3, 1×1}

In MCH, based on References [13, 14], the filter size f and number of feature maps n at $K\times$ magnification were set as shown in Table 2.7. In the 3Path-CNN, filter size f was set as shown in Table 2.8. The number of feature maps n_1 and n_2 of the 3Path-CNN are set according to the best result in Section 2.4.3.

The training images, number of backpropagations, learning rate, and evaluation methods for the CNNs are the same as in Section 2.4.1. The testing datasets were Set5 [30], Set14 [31], and BSDS100 [29].

Tables 2.9 and 2.10 show the PSNRs and SSIMs for Set5 and Set14 at 2× magnification. Tables 2.11, 2.12, and 2.13 show the average PSNR and SSIM on BSDS100 at 2×, 3×, and 4× magnification, respectively.

For all testing datasets, the 3Path-CNN had similar or slightly lower PSNRs and SSIMs compared with MCH. This is because the number of parameters for the 3Path-CNN was less than that of MCH.

Moreover, compared with MCH, the processing times for the 3Path-CNN (8-24+8+1-4) and 3Path-CNN (4-28+4+1-4) reduced to approximately 25% and 20%, respectively. Tables 2.14 and 2.15 show the total number of internal parameters for MCH and 3Path-CNN (4-28+4+1-4). The total number of parameters is related to computational cost. Compared with the conven-

Table 2.9: PSNRs, SSIMs, and processing times for 2× mangification (Set5).

Testing image	PSNR [dB] / SSIM / Processing time [s]			
	Bicubic	MCH [14]	3Path-CNN (8-24+8+1-4)	3Path-CNN (4-28+4+1-4)
baby	36.95 / 0.9516 / 0.0008	38.29 / 0.9639 / 0.4776	38.32 / 0.9637 / 0.1178	38.27 / 0.9633 / 0.0900
bird	36.81 / 0.9720 / 0.0005	41.15 / 0.9857 / 0.2227	41.21 / 0.9858 / 0.0595	40.90 / 0.9849 / 0.0425
butterfly	27.49 / 0.9153 / 0.0005	33.18 / 0.9661 / 0.1997	32.93 / 0.9647 / 0.0539	32.77 / 0.9635 / 0.0384
head	34.74 / 0.8623 / 0.0005	35.60 / 0.8854 / 0.2171	35.65 / 0.8862 / 0.0580	35.60 / 0.8855 / 0.0419
Woman	32.30 / 0.9468 / 0.0005	35.45 / 0.9680 / 0.2168	35.33 / 0.9675 / 0.0589	35.36 / 0.9671 / 0.0412
Average	33.66 / 0.9296 / 0.0006	**36.73** / **0.9538** / 0.2668	36.69 / 0.9536 / 0.0696	36.58 / 0.9529 / **0.0508**

Bold red font indicates the best in all super-resolution methods.

Table 2.10: PSNRs, SSIMs, and processing times for 2× mangification (Set14).

Testing image	PSNR [dB] / SSIM / Processing time [s]			
	Bicubic	MCH [14]	3Path-CNN (8-24+8+1-4)	3Path-CNN (4-28+4+1-4)
baboon	24.86 / 0.6977 / 0.0008	25.72 / 0.7700 / 0.4429	25.72 / 0.7694 / 0.1116	25.68 / 0.7675 / 0.0864
Barbara	27.91 / 0.8401 / 0.0011	28.34 / 0.8714 / 0.7557	28.37 / 0.8711 / 0.1897	28.38 / 0.8727 / 0.1533
bridge	26.63 / 0.7932 / 0.0008	27.92 / 0.8518 / 0.4670	27.93 / 0.8520 / 0.1178	27.90 / 0.8513 / 0.0893
coastguard	29.22 / 0.7879 / 0.0006	30.55 / 0.8464 / 0.2499	30.57 / 0.8476 / 0.0653	30.59 / 0.8481 / 0.0471
comic	25.80 / 0.8465 / 0.0006	28.34 / 0.9155 / 0.2343	28.24 / 0.9145 / 0.0625	28.24 / 0.9139 / 0.0445
face	34.71 / 0.8621 / 0.0005	35.57 / 0.8855 / 0.2177	35.62 / 0.8860 / 0.0605	35.57 / 0.8855 / 0.0435
flowers	30.16 / 0.8978 / 0.0006	33.18 / 0.9367 / 0.3608	33.07 / 0.9362 / 0.0912	32.95 / 0.9353 / 0.0696
foreman	35.43 / 0.9534 / 0.0005	39.19 / 0.9704 / 0.2489	39.18 / 0.9702 / 0.0652	38.96 / 0.9695 / 0.0472
lenna	34.55 / 0.9111 / 0.0008	36.46 / 0.9288 / 0.4707	36.43 / 0.9282 / 0.1173	36.40 / 0.9282 / 0.0892
man	29.14 / 0.8453 / 0.0008	30.87 / 0.8873 / 0.4693	30.84 / 0.8870 / 0.1172	30.79 / 0.8858 / 0.0891
monarch	32.73 / 0.9599 / 0.0010	37.65 / 0.9765 / 0.7088	37.48 / 0.9762 / 0.1765	37.30 / 0.9758 / 0.1407
pepprer	35.00 / 0.9071 / 0.0008	37.07 / 0.9209 / 0.4708	37.03 / 0.9205 / 0.1185	36.96 / 0.9200 / 0.0900
ppt3	26.64 / 0.9438 / 0.0009	30.17 / 0.9737 / 0.6883	29.94 / 0.9743 / 0.1604	30.00 / 0.9718 / 0.1312
zebra	30.46 / 0.9089 / 0.0007	33.58 / 0.9418 / 0.4234	33.56 / 0.9420 / 0.1070	33.46 / 0.9413 / 0.0797
Average	30.23 / 0.8682 / 0.0008	**32.47** / **0.9055** / 0.4437	32.43 / 0.9054 / 0.1115	32.37 / 0.9048 / **0.0858**

Bold red font indicates the best in all super-resolution methods.

Table 2.11: PSNRs, SSIMs, and processing times for 2× mangification (BSDS100).

Testing image	PSNR [dB] / SSIM / Processing time [s]			
	Bicubic	MCH [14]	3Path-CNN (8-24+8+1-4)	3Path-CNN (4-28+4+1-4)
Average	29.45 / 0.8423 / 0.0006	**31.21** / **0.8867** / 0.3212	31.18 / 0.8863 / 0.0824	31.15 / 0.8859 / **0.0615**

Bold red font indicates the best in all super-resolution methods.

tional MCH, 3Path-CNN reduced the number of parameters relative to performance with the multiple-path architecture. Consequently, 3Path-CNN reduced the number of multiplications to approximately 7.2%, compared with MCH. This is a reason for the reduced processing time.

Table 2.12: PSNRs, SSIMs, and processing times for 3× mangification (BSDS100).

Testing image	PSNR [dB] / SSIM / Processing time [s]			
	Bicubic	MCH [14]	3Path-CNN (16-48+16+1-9)	3Path-CNN (8-56+8+1-9)
Average	27.13 / 0.7380 / 0.0006	**28.31** / **0.7849** / 0.1799	28.25 / 0.7830 / 0.0293	28.22 / 0.7816 / **0.0254**

Bold red font indicates the best in all super-resolution methods.

Table 2.13: PSNRs, SSIMs, and processing times for 4× mangification (BSDS100).

Testing image	PSNR [dB] / SSIM / Processing time [s]			
	Bicubic	MCH [14]	3Path-CNN (16-48+16+1-16)	3Path-CNN (8-56+8+1-16)
Average	25.91 / 0.6679 / 0.0006	**26.76** / 0.7042 / 0.1343	**26.76** / **0.7074** / 0.0192	26.73 / 0.7059 / **0.0167**

Bold red font indicates the best in all super-resolution methods.

Table 2.14: Number of internal parameters of MCH (2× magnification).

Index	first layer	second layer	thrid layer
Filter size f	5×5	3×3	3×3
Number of feature maps n	64	32	4
Number of weights W	1,600 (1×5×5×64)	51,200 (64×5×5×32)	1,152 (32×3×3×4)
Total number of weights W_t	53,952 (1,600 + 51,200 + 1,152)		

Figures 2.17, 2.18, and 2.19 show examples of the magnified images. Both MCH and 3Path-CNN magnified edges more sharply than Bicubic interpolation. The subjective evaluation result was roughly equal to the objective results shown in Tables 2.9, 2.10, 2.11, 2.12, and 2.13. Meanwhile, MCH and 3Path-CNN differed slightly in PSNR; however, no significant differences were observed subjectively. These results show that 3Path-CNN maintains the image quality subjectively, compared with MCH.

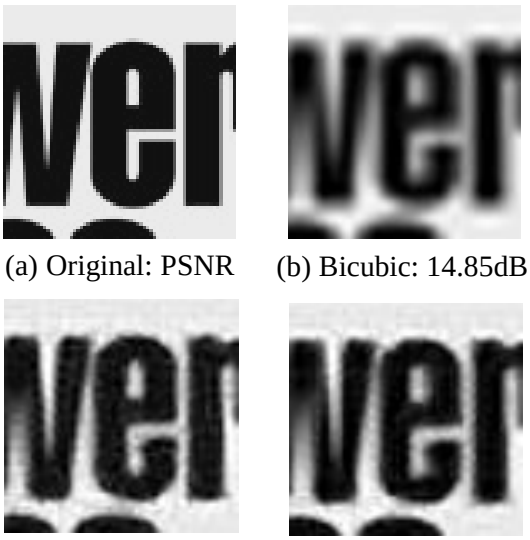
Comparison with Other High-Performance Super-Resolution Methods

This section compares the proposed method, conventional MCH [14], and the following other high-performance super-resolution methods: A+ [5], SRCNN [3], NBSRF [6], PSyCo [7], FSRCNN [11], and MCH-HV [15]. Figures 2.20, 2.21, and 2.22 show the average PSNR results on BSDS100 at 2×, 3×, and 4× magnification, respectively.

First, this section focuses on the three example-based super-resolutions: A+, NBSRF, and PSyCo. NBSRF and PSyCo were better than A+ in both image quality and processing time.

Table 2.15: Number of internal parameters of the 3Path-CNN (2× magnification).

Index	first layer	second layer	thrid layer
Filter size f	5×5	3×3	3×3
Number of feature maps n	4	28	4
Number of weights W	100 (1×5×5×4)	2,800 (4×5×5×28)	1,008 (28×3×3×4)
Total number of weights W_t	3,908 (100 + 2,800 + 1,008)		



(a) Original: PSNR (b) Bicubic: 14.85dB

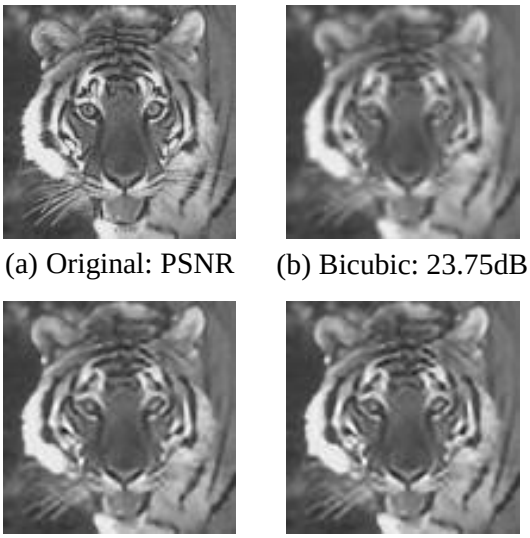
(c) MCH: 18.76dB (d) 3Path-CNN: 17.63dB

Figure 2.17: Part of "108005" from BSDS100 (2× magnification).

NBSRF and PSyCo were the best at balancing between PSNR and processing time. They were better than CNN-based super-resolutions: SRCNN and MCH. The conventional single-path CNN seems inefficient.

This section focuses on the five CNN-based methods. The processing times were shorter in the order of SRCNN, MCH, MCH-HV, FSRCNN, and 3Path-CNN. In particular, the 3Path-CNN had the same or higher processing time compared with that of NBSRF and achieved approximately 2.5- and 5-times faster processing times at 3× and 4× magnification, respectively. 3Path-CNN achieved a PSNR lower than that of NBSRF by approximately 0.05 dB. However, in balancing image quality and processing time, 3Path-CNN achieved a good score.

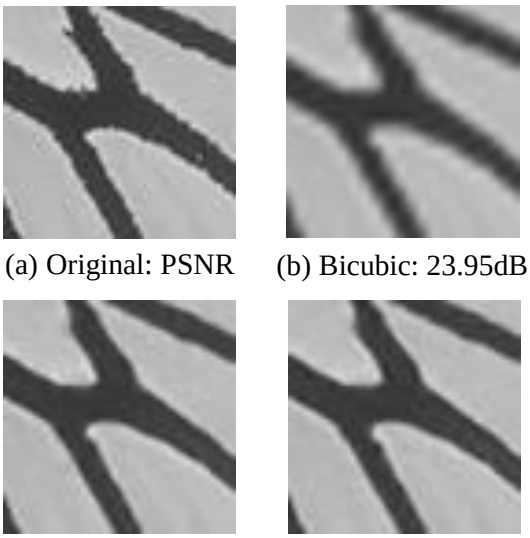
From these evaluation results, the proposed CNN architecture can improve the practicality of



(a) Original: PSNR (b) Bicubic: 23.75dB

(c) MCH: 25.59dB (d) 3Path-CNN: 25.58dB

Figure 2.18: Part of "butterfly" from Set5 (3× magnification).



(a) Original: PSNR (b) Bicubic: 23.95dB

(c) MCH: 29.45dB (d) 3Path-CNN: 28.98dB

Figure 2.19: Part of "ppt" from Set14 (4× magnification).

CNN-based super-resolutions.

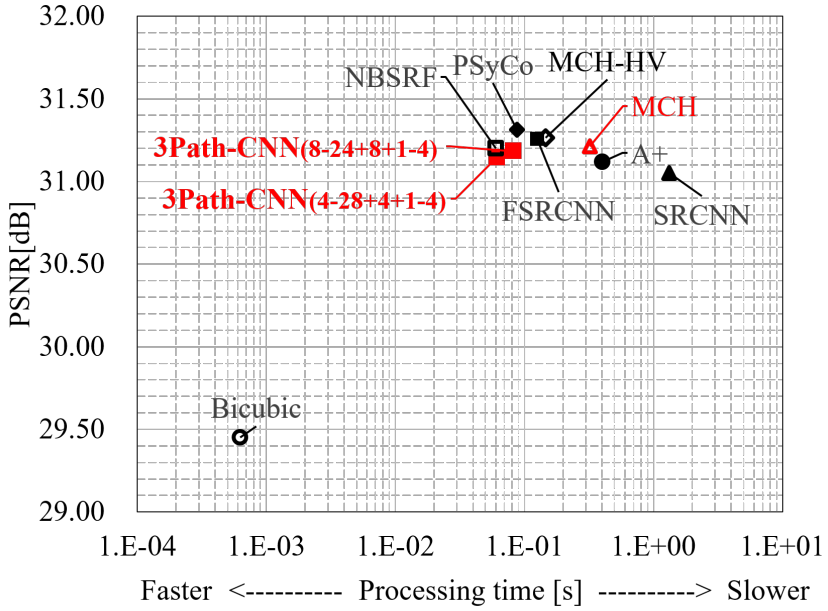


Figure 2.20: PSNRs and processing times (BSDS100, 2× magnification). The horizontal and vertical axes are the processing time and average PSNR, respectively.

2.5 Conclusion

This chapter proposed super-resolutions using a CNN with multiple propagation paths to achieve high-quality images and low computation costs. In the proposed method, a CNN had multiple paths from low to deep layers and generated low- and high-frequency signals independently. Based on this, the number of parameters relative to the image quality was reduced. In addition, the step learning separated the frequency band. In the evaluation results, the processing time of the proposed method was reduced to approximately a quarter of that of conventional MCH with the same PSNR. These results show that the proposed architecture with multiple propagation paths is useful. The proposed method helps introduce CNN-based super-resolutions to televisions and smartphones without accelerators, such as GPUs. However, in CNN-based super-resolution, the computation cost are typically larger than linear interpolations or example-based super-resolutions in terms of the PSNR relative to processing time. The implementation for efficient architecture with a lower computation cost for image quality is the focus of future work.

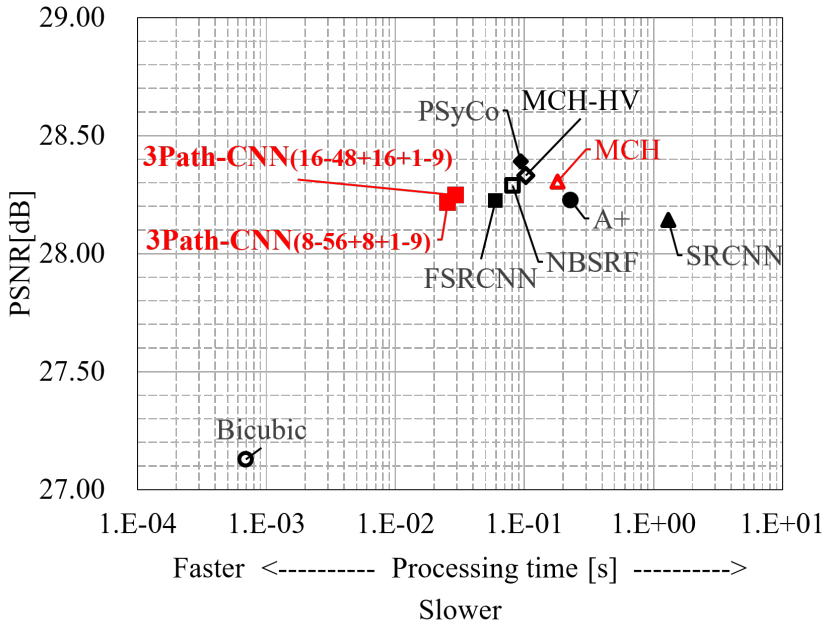


Figure 2.21: PSNRs and processing times (BSDS100, 3× magnification). The horizontal and vertical axes are the processing time and average PSNR, respectively.

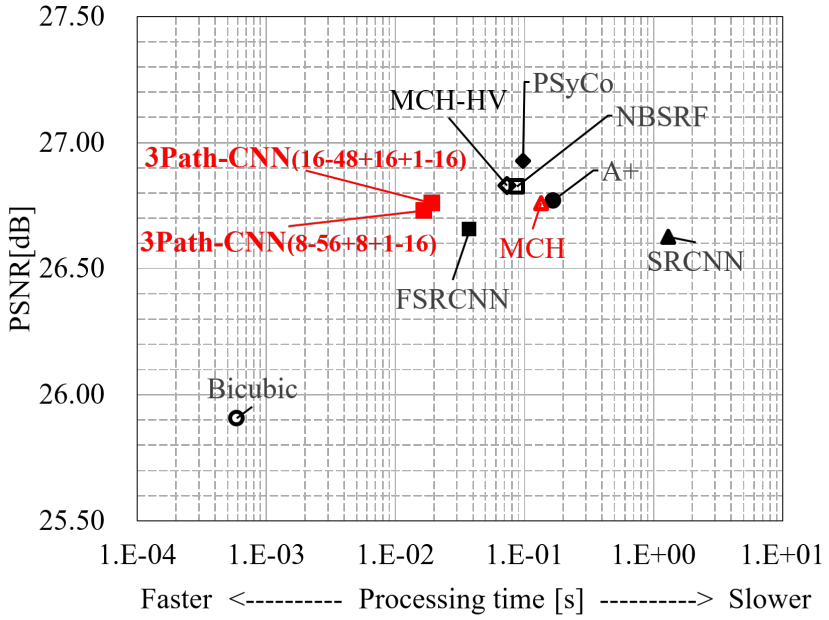


Figure 2.22: PSNRs and processing times (BSDS100, 4× magnification). The horizontal and vertical axes are the processing time and average PSNR, respectively.

3 Luminance Isotropy Improvement for Convolutional Neural Network-Based Image Super-Resolution

3.1 Introduction

Image upsampling techniques are used in a number of applications and devices such as digital cameras, smart phones, and televisions. Recently, many image super-resolution techniques based on CNNs have achieved state-of-the-art results [3, 14, 32, 33]. However, the CNNs do not necessarily show the same characteristics for all luminance ranges because the CNN itself is a nonlinear function and its parameters are initialized with random values. This chapter discovered that conventional super-resolution CNN outputs different responses from positive and negative impulse signals as proved later in Section 3.3.

On the other hand, the luminance distributions of benchmark datasets for super-resolution [28–31, 34–36] are different from each other, as shown in Figure 3.1. The performance of the CNN is dependent on the characteristic of the training dataset. For practical use, however, the luminance distributions of training and inferred images are not always the same. These differences may cause performance degradation. Thus, the method which achieves luminance isotropy is one of the promising strategies for improving the performance of CNN-based super-resolution.

This chapter propose two methods. One is "luminance inversion training." This method augments the dataset with the luminance inversion of the original dataset; therefore, it gives luminance variability and isotropy to the training dataset. The other method is "luminance inversion averaging." This method gives complete luminance isotropy to the CNN-based super-resolution. Specifically, the CNN generates two magnified images from a positive image and a luminance inversed image (negative image), and then the final magnified image is obtained from the average of the positive and re-inversed negative images. These proposed methods improve luminance isotropy and are promising strategies for improving the performance of CNN-based super-resolution.

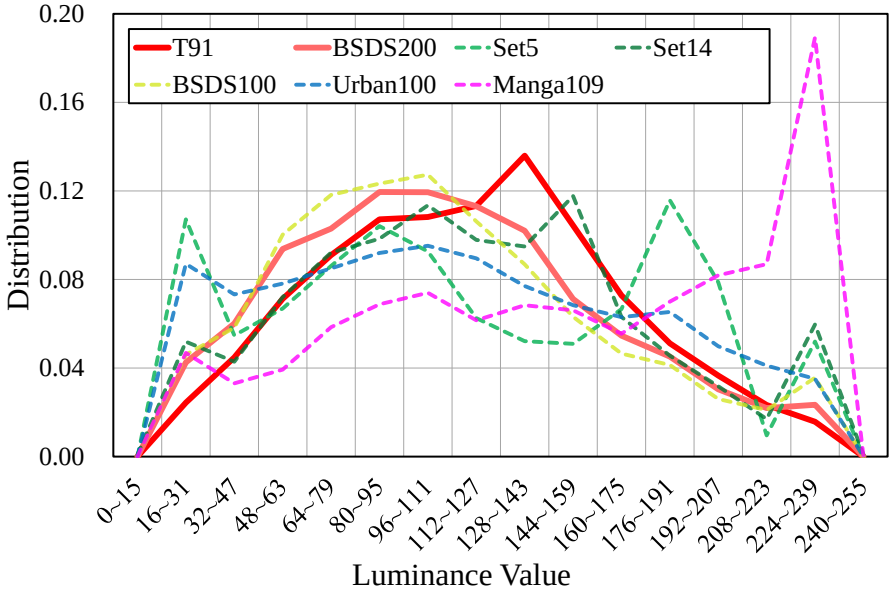


Figure 3.1: Luminance distributions of benchmark datasets for super-resolution. The datasets of solid lines and broken lines are often used as training images and testing images, respectively.

3.2 Proposed Methods

3.2.1 Luminance Inversion Training

Data augmentation is one of the most widely used techniques for boosting performance with deep learning. For image super-resolution, some useful augmentations, such as random cropping, flipping, scaling, rotating, and color jittering have already been proposed [33]. This chapter proposes a new data augmentation method for super-resolution, called luminance inversion training (LIT). Let $\boldsymbol{Y}$ be the 8-bit image with luminance channel of YCrCb space. In this method, the original image $\boldsymbol{Y}$ is augmented by luminance inversion as follows:

$$\overline{\boldsymbol{Y}} = 255 - \boldsymbol{Y}. \tag{3.1}$$

Then, $\overline{\boldsymbol{Y}}$ is added to the original dataset. This method gives the luminance variability and isotropy to the training dataset.

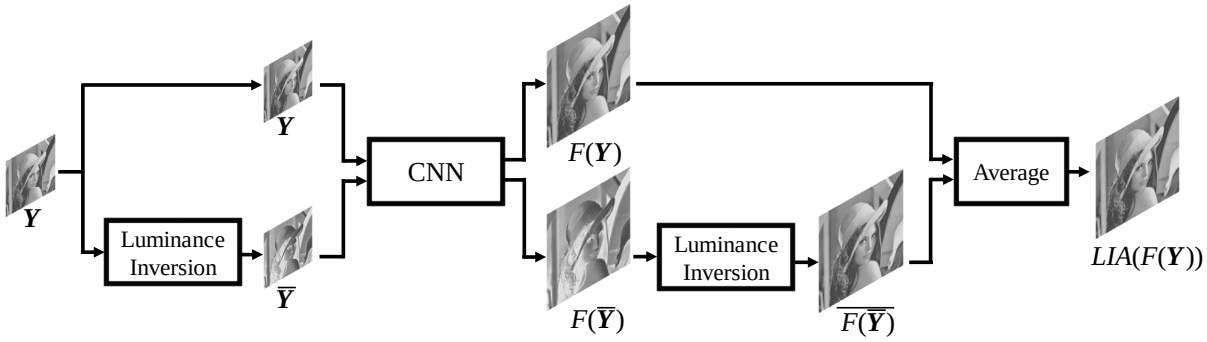


Figure 3.2: Luminance inversion averaging.

3.2.2 Luminance Inversion Averaging

This section proposes another method, called luminance inversion averaging (LIA) for complete luminance isotropy. Figure 3.2 shows the flow of this method. First, the positive image $\boldsymbol{Y}$ and the negative image $\overline{\boldsymbol{Y}}$ are prepared. The negative image $\overline{\boldsymbol{Y}}$ is calculated using the luminance inversion of the positive image as follows:

$$\overline{\boldsymbol{Y}} = 255 - \boldsymbol{Y}. \tag{3.2}$$

Thereafter, the magnified images of the positive image $\boldsymbol{Y}$ and negative image $\overline{\boldsymbol{Y}}$ are generated as follows:

$$F(\boldsymbol{Y}) = (\boldsymbol{Y}) \uparrow_s, \tag{3.3}$$

$$F(\overline{\boldsymbol{Y}}) = (\overline{\boldsymbol{Y}}) \uparrow_s, \tag{3.4}$$

respectively, where $\uparrow_s$ is the CNN-based super-resolution F with scale factor s. Next, the magnified image from the negative image $F(\overline{\boldsymbol{Y}})$ is re-inversed as follows:

$$\overline{F(\overline{\boldsymbol{Y}})} = 255 - F(\overline{\boldsymbol{Y}}). \tag{3.5}$$

Finally, the output image is calculated using the pixel average of $F(\boldsymbol{Y})$ and $\overline{F(\overline{\boldsymbol{Y}})}$ as follows:

$$LIA(F(\boldsymbol{Y})) = \frac{F(\boldsymbol{Y}) + \overline{F(\overline{\boldsymbol{Y}})}}{2}. \tag{3.6}$$

$LIA(F(\boldsymbol{Y}))$ is always isotropy for the luminance signal like linear interpolation. Note that the $LIA(*)$ is an external processing of the $F(\boldsymbol{Y})$; therefore, LIA is applicable for any type of CNN-based super-resolution technique.

Table 3.1: Conditions of CNN-based super-resolutions.

Method	Convolutional layers	Training dataset
MCH [14]	3	T91 [28]
LapSRN [32]	14	T91 [28] + BSDS200 [29]

Table 3.2: PSNRs and SSIMs of 2× magnified images for all testing datasets.

Method	PSNR [dB] / SSIM	
	MCH	LapSRN
Baseline	32.25 / 0.9176	32.48 / 0.9208
+Luminance inversion training (LIT)	32.36 / 0.9184	32.53 / 0.9208
+Luminance inversion averaging (LIA)	**32.45** / **0.9193**	**32.63** / **0.9216**

Bold red font indicates the best scores in each CNN-based super-resolution.

3.3 Evaluation Experiments

This section proves the effectiveness of the proposed methods by applying them to the CNN-based super-resolutions called MCH [14] and Laplacian pyramid super-resolution network (LapSRN) [32] shown in Table 3.1. MCH is developed for low-complexity and high-speed processing, and its hyperparameters are the same as the original study [14], excluding the fact that the number of backpropagations is 7.5 million. LapSRN is developed for the state-of-the-art image-quality performance, and its hyperparameters are the same as the original study [32] excluding the fact that there is no data augmentation. MCH is implemented with a Caffe package [22]. LapSRN is implemented with a MatConvNet package [37] and the package distributed by the original study [32]. Testing datasets are Set5 [30], Set14 [31], BSDS100 [29], Urban100 [34], and Manga109 [35, 36]. This experiment focuses only on the luminance channel in the YCrCb space and evaluate 2× image magnification. LIT doubles the original training dataset; however, no other data augmentations are used for training and testing images.

3.3.1 Objective Evaluation with PSNR and SSIM.

Table 3.2 shows the average PSNR and SSIM for all testing datasets. PSNR and SSIM are calculated with the same as Equation 2.15 and Equation 2.16, respectively. Each proposed method improved PSNR in both MCH and LapSRN. Especially, LIA is effective and improved

Table 3.3: PSNRs and SSIMs of 2× magnified images with MCH.

Method	PSNR[dB] / SSIM				
	Set5	Set14	BSDS100	Urban100	Manga109
Baseline	36.73 / 0.9539	32.47 / 0.9057	31.21 / 0.8869	29.41 / 0.8951	35.56 / 0.9663
+Luminance inversion training (LIT)	36.80 / 0.9542	32.52 / 0.9062	31.25 / 0.8874	29.46 / 0.8960	35.83 / 0.9672
+Luminance inversion averaging (LIA)	**36.88** / **0.9549**	**32.59** / **0.9071**	**31.29** / **0.8882**	**29.52** / **0.8971**	**35.98** / **0.9683**

Bold red font indicates the best score in each testing dataset.

Table 3.4: PSNRs and SSIMs of 2× magnified images with LapSRN.

Method	PSNR[dB] / SSIM				
	Set5	Set14	BSDS100	Urban100	Manga109
Baseline	36.86 / 0.9558	32.56 / 0.9082	31.44 / 0.8899	29.60 / 0.8989	35.86 / 0.9693
+Luminance inversion training (LIT)	36.85 / 0.9554	32.55 / 0.9081	31.43 / 0.8896	29.62 / 0.8988	36.01 / 0.9695
+Luminance inversion averaging (LIA)	**36.97** / **0.9562**	**32.62** / **0.9088**	**31.47** / **0.8903**	**29.68** / **0.8998**	**36.20** / **0.9703**

Bold red font indicates the best score in each testing dataset.

the average PSNR by 0.15–0.20 dB.

Tables 3.3 and 3.4 show the average PSNR for each testing dataset. First, this section focuses on LIT. Although greatly improving the average PSNRs in the case of Manga109, this method only slightly improved or degraded them in the other datasets. Thus, the LIT does not necessarily improve the image quality.

Next, this section focuses on LIA. This method improved the PSNRs in all testing datasets. These results mean that the internal improvement of the CNN is difficult; however, the LIA of the external processing can easily improve the super-resolution performance.

Finally, this section discusses the relationship between the luminance distributions of datasets and the improvements of image quality. Figure 3.1 shows that the luminance distribution of BSDS100 is similar to other distributions used for training CNNs; however, luminance distribution of Manga109 is different from them. In such a case, LIA is especially effective. Table 3.3 shows that LIA improved the PSNR by only 0.08 dB in BSDS100 but by 0.42 dB in Manga109. Table 3.4 also shows the same tendency. From these results, LIA is greatly effective for correcting the mismatch between the luminance distributions of training and inferred images.

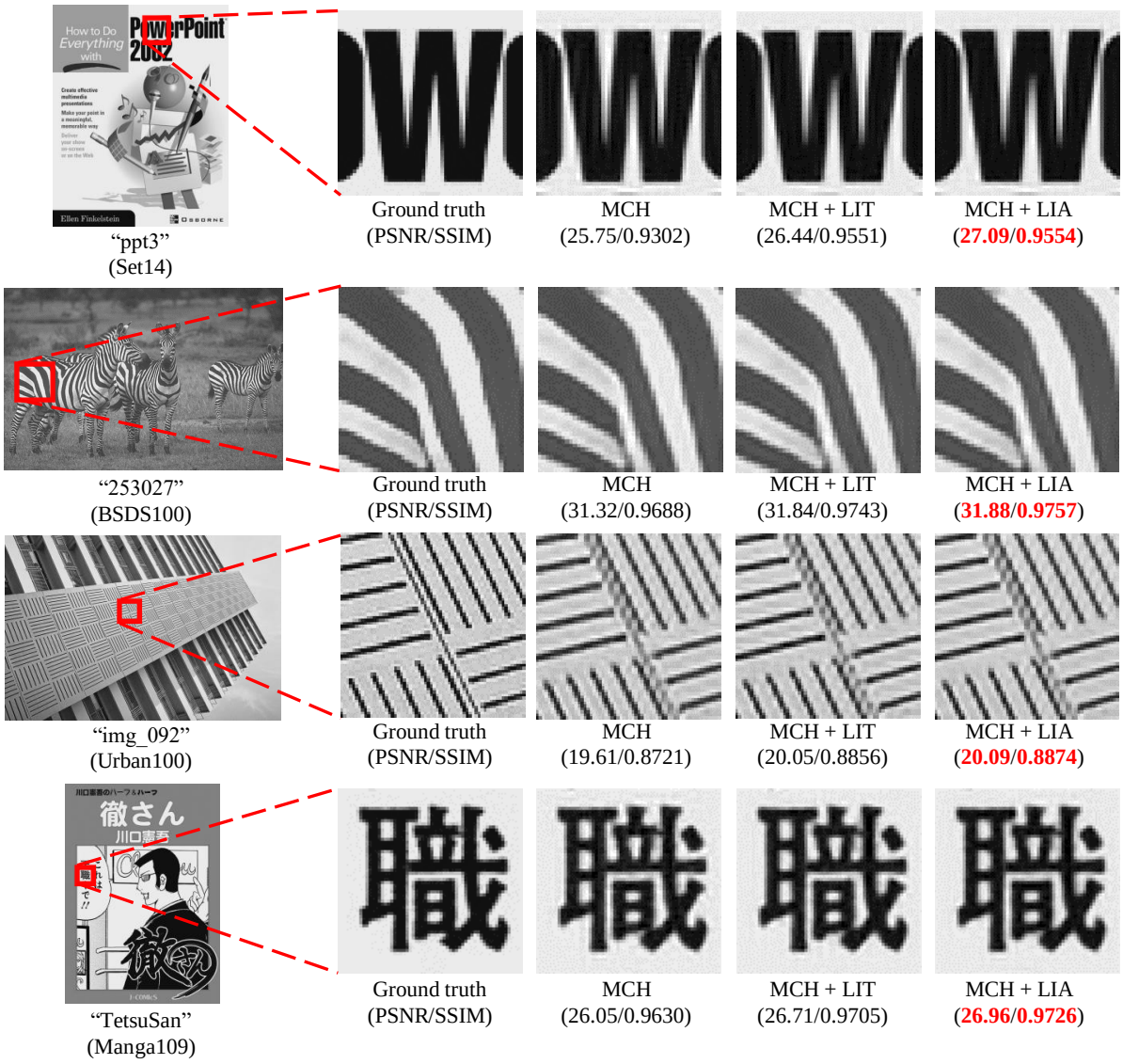
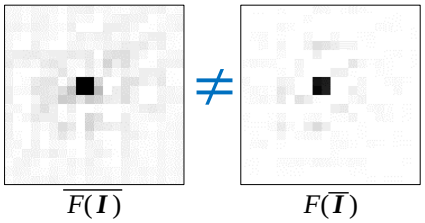


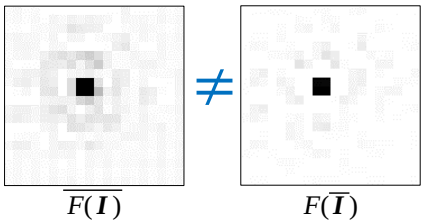
Figure 3.3: Visual comparison on 2× magnified images with MCHs. Bold red font indicates the best score in each testing image.

3.3.2 Subjective Evaluation of Magnified Images.

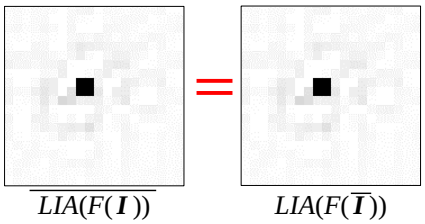
Figure 3.3 shows the magnified images with MCHs. LIA reduces the overshoots around the edge and outlines these edges clearly, compared with other methods. This is because LIA calculates the average of the positive and negative output signals, and then their noises are canceled.



(a) MCH.



(b) MCH + LIT.



(c) MCH + LIA.

Figure 3.4: Isotropic of Impulse response with MCHs. $\boldsymbol{I}$ and $\overline{\boldsymbol{I}}$ denote positive and negative impulse signals, respectively.

3.3.3 Isotropic of Impulse Response

Figure 3.4 shows the inversed positive and negative impulse responses. The conventional MCH has no luminance isotropy. In LIT, although the training dataset has luminance isotropy, the impulse responses have no luminance isotropy. This is because the CNN itself is a nonlinear function and its parameters are initialized with random values. On the other hand, the inversed positive and negative impulse responses are exactly equal in LIA. Thus, LIA has the perfect isotropy for the luminance signal. This indicates that the super-resolution gives the same performance in both bright and dark areas. The authors believe that this is one of the important characteristics that super-resolutions should have.

3.4 Conclusion

In this chapter, "luminance inversion training (LIT)" and "luminance inversion averaging (LIA)" were proposed to improve the luminance isotropy of CNN-based image super-resolutions. Experimental results have shown that the average PSNR for super-resolution using LIA is approximately 0.15–0.20 dB higher than that for the conventional super-resolution. This chapter has ensured that the proposed method can improve the performances of CNN-based super-resolutions even if the luminance distributions of training and inferred images are different from each other. Additionally, the proposed methods are applicable for any CNN-based super-resolution. However, LIA does not improve the internal structure of CNN and requires twice the processing time in principle. The implementation for internal isotropy of CNN will be the focus of the future work.

4 Multi-Category Image Super-Resolution with Convolutional Neural Network and Multi-Task Learning

4.1 Introduction

Image upsampling techniques are used in several applications and devices such as digital cameras, smart phones, and televisions. When inputting a low-resolution image to a high-resolution display device, some resolution conversion is required. The image quality can become blurry and artifacting using conventional interpolation methods such as Bicubic interpolation. According to the sampling theorem, signals that exceed the Nyquist frequency (hereinafter, referred to as high-frequency components) are not included in the original signal. Because Bicubic interpolation is a technique to expand signals below the Nyquist frequency, high-frequency components are not generated.

In contrast, the method called "super-resolution" has been intensively investigated recently. The super-resolution method generates a high-resolution image by estimating high-frequency components not included in input signals. The SRCNN, proposed by Dong et al. [2,3], improves image qualities significantly. After SRCNN was proposed, several super-resolutions with CNNs have been proposed [2, 3, 10, 11, 13, 14, 23, 24, 26, 27, 33, 38].

However, most CNN-based super-resolutions are trained for a limited image category. For example, the MCH [14] was trained with the T91 dataset [28] of natural images and then applied to the Set5 [30] and Set14 [31] datasets in the same category. In practical use, however, users can input several types of images. Figure 4.1 shows examples of a natural image from BSDS100 [29], a manga image from Manga109 [35, 36], and a text image from Hradiš's dataset [39]. They have different features from each other. If a CNN is trained for a specific category, it often causes performance degradation for the other categories. Table 4.1 presents the image qualities obtained by the three MCHs that are trained with different categories. When the training and testing image categories are the same, the MCH yields the highest image quality in each image

Table 4.1: PSNRs of 2× magnified images with the super-resolution CNN trained with each image category.

Training images	Testing images		
	Natural images	Manga images	Text images
Natural images	**31.31**	35.48	23.25
Manga images	31.20	**36.65**	24.53
Text images	28.08	29.77	**24.66**

Bold red font indicates the best score in each testing dataset.

category. The MCH trained with natural images exhibits a good PSNR of 31.31 dB for natural testing images, whereas the MCH trained with manga images or text images yields unsatisfactory PSNRs of 31.20 dB or 28.08 dB for natural testing images, respectively. Similarly, MCH trained with manga images shows the highest PSNR of 36.65 dB for manga testing images, and MCH trained with text images shows the highest PSNR of 24.66 dB for text testing images. Therefore, the user can determine if the CNN achieves a good result for a previously trained category. If a super-resolution for multiple image categories with conventional CNN is required, the possible solutions are as follows:

PlanA Train multiple CNNs with every image category;

PlanB Train single CNN with all image categories.

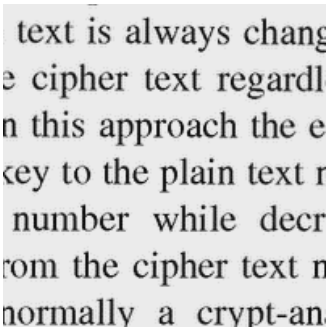
In *PlanA*, as shown in Figure 4.2-(a), the CNNs are trained for each image category. In particular, super-resolution techniques for specific category have been studied [23]. However, *PlanA* must estimate the category of the input image manually or automatically to select a suitable CNN. If users know the input image category, they need to select the optimal CNN manually and determine in advance the type of images used for learning the CNN. In addition, the solution requires maintaining the parameters of multiple CNNs. Therefore, *PlanA* is not suitable for low-end devices with a small memory capacity.

In *PlanB*, as shown in Figure 4.2-(b), a single CNN is trained with images of all categories. However, because the CNN has difficulty in understanding the image categories, it operates uniformly for all inputs. In this case, although the CNN can show better results for all categories, it cannot yield the best result for each.

Moreover, [16,33,38] introduced some indicators for improving image quality performance after

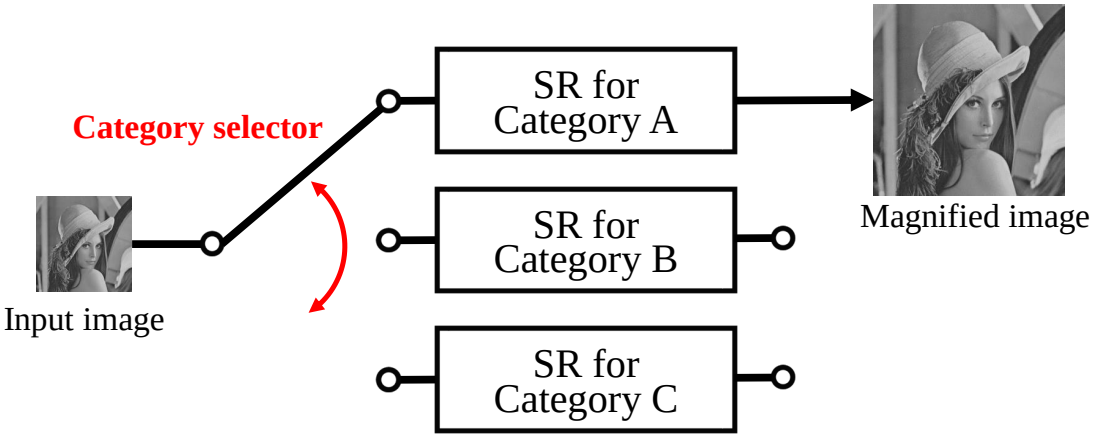
(a) Example of natural image from BSDS100.

(b) Example of manga image from Manga109.

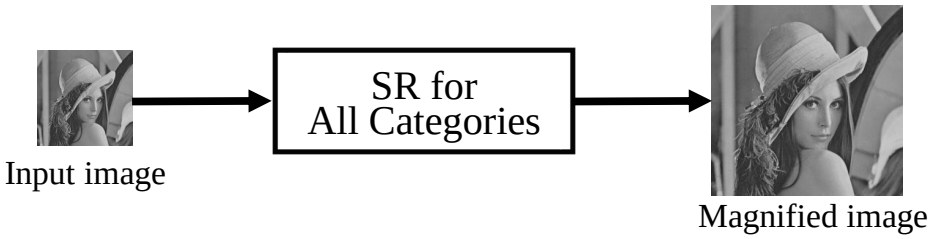


(c) Example of text image from Hradiš's dataset.

Figure 4.1: Visual difference between multiple categories.



(a) *PlanA* (Train multiple CNNs with every image category).



(b) *PlanB* (Train single CNN with all image categories).

Figure 4.2: How to improve the image qualities of multiple categories.

investigating recent super-resolution techniques. A promising strategy is to learn the potential correlations among image characteristics and reflect it in the loss function. Although the mean square error is widely used for the loss function between low-resolution and high-resolution images, it does not represent the object information of the image. Thus, the CNN is trained uniformly without understanding the object.

Moreover, multi-task learning (MTL) has been studied in recent deep learning [25, 33, 40–43]. MTL is a technique that manages multiple tasks with single machine learning model and improves main-task performance using the optional information contained in sub-tasks [40]. In CNN-based super-resolutions, MTL is implemented by combining multiple loss functions [25,43]. Shi et al. improved the image quality by adding the optional information about object boundaries on sub-networks [25]. Rad et al. proposed an architecture that can generates the super-resolution

image and the semantic information of natural image [30, 31, 43, 44]. Reference [33] stated that because different tasks focus on different aspects of the data, combining related tasks with super-resolution models usually improves the performance of the super-resolution by providing additional information and knowledge.

Based on this section's survey, this chapter considered the training for understanding image categories to be a promising solution for improving the image quality performance. Thus, this chapter proposes an image super-resolution method that is adaptable to multiple image categories, implemented with a single CNN and MTL. The proposed architecture has two parallel outputs: a magnified image and its category. In the proposed architecture, the main CNN is equivalent to the conventional super-resolution CNN. Moreover, an optional neural network (NN) branches out from the middle layer of the main CNN. The branched NN is used to learn the image categories in the training phase. This is removed in the inference phase. This removable architecture can make an internal category classifier in the super-resolution CNN. Because the trained CNN has the category classification ability, users do not need to input the image category manually. Therefore, the proposed method can address the problems of *PlanA* and *PlanB*.

The remainder of this chapter is organized as follows. Section 4.2 introduces the multi-category image super-resolution with CNN and MTL. Section 4.3 presents experiments to show the usefulness of the proposed method.

4.2 Multi-Category Image Super-Resolution with Convolutional Neural Network and Multi-Task Learning

In this section, we propose an image super-resolution technique that is adaptable to multiple image categories, implemented with a single CNN and MTL. Figure 4.3 shows the concept of the proposed method. This architecture has an internal category classifier; however, it has no input or output for the category information. The category information is estimated internally and only used for optimizing the super-resolution.

The remainder of this section is organized as follows. Section 4.2.1 presents the CNN architecture of the proposed method. Section 4.2.2 explains the training method of the proposed method. Finally, Section 4.2.3 details the behavior of the proposed method in inference. In the following, this chapter refers to the proposed method as multi-channel convolutional neural

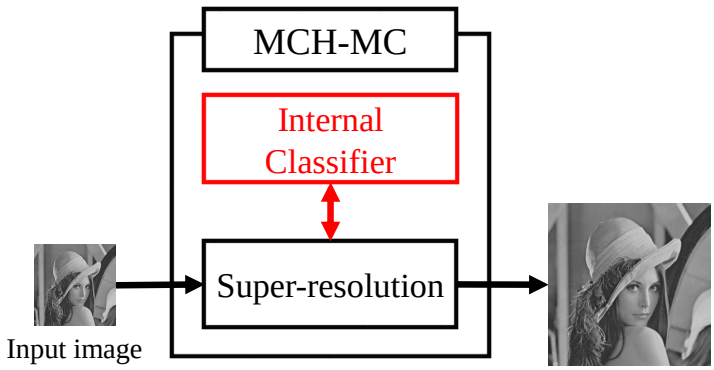


Figure 4.3: Concept of the proposed architecture for multiple categories.

network for multi-category (MCH-MC).

4.2.1 Proposed CNN Architecture

Figure 4.4 shows the proposed CNN architecture. Let $\boldsymbol{Y}$ and K be the low-resolution image and magnification ratio, respectively. The output from the first to third convolution layers is given by the same approach as MCH as shown in Section 2.2.1. The outputs of the first convolution layer are calculated as follows:

$$F_1(\boldsymbol{Y}) = \max(0, W_1 * \boldsymbol{Y} + B_1), \tag{4.1}$$

where W_1 is the filter for convolution, B_1 is the bias, and "$*$" denotes the convolution operation. W_1 is 4-dimensional tensor, and its size is $1 \times f_1 \times f_1 \times n_1$, where $f_1 \times f_1$ is the filter size, and n_1 is the number of filters in the first convolution layer. $\max(0, x)$ denotes the ReLU [18]. The outputs of the second convolution layer are calculated as follows:

$$F_2(\boldsymbol{Y}) = \max(0, W_2 * F_1(\boldsymbol{Y}) + B_2). \tag{4.2}$$

The size of W_2 is $n_1 \times f_2 \times f_2 \times n_2$, where $f_2 \times f_2$ is the filter size, and n_2 is the number of filters in the second convolution layer. Finally, the K^2 pixels are calculated as follows:

$$F_3(\boldsymbol{Y}) = W_3 * F_2(\boldsymbol{Y}) + B_3. \tag{4.3}$$

where $F_3(\boldsymbol{Y})$ has K^2 channels. The size of W_3 is $n_2 \times f_3 \times f_3 \times K^2$. Let $F_{3.1}(\boldsymbol{Y})$, $F_{3.2}(\boldsymbol{Y})$, $\cdots$, $F_{3.K^2}(\boldsymbol{Y})$ be separate channels of $F_3(\boldsymbol{Y})$. Thereafter, super-resolution image $F(\boldsymbol{Y})$ is generated by synthesizing each channel $F_{3.1}(\boldsymbol{Y})$, $F_{3.2}(\boldsymbol{Y})$, $\cdots$, $F_{3.K^2}(\boldsymbol{Y})$ as shown in Equation 2.4.

In contrast, image classification NN is branched from the second feature map, $F_2(\boldsymbol{Y})$. Global average pooling (GAP) is a technique that outputs the average pixel value of each feature map [45]. Using GAP, one-dimensional vector $F_4(\boldsymbol{Y})$ of length n_2 is calculated as follows:

$$F_4(\boldsymbol{Y}) = \mathrm{GAP}(F_2(\boldsymbol{Y})), \tag{4.4}$$

where $\mathrm{GAP}(F_2(\boldsymbol{Y}))$ outputs the pixel averages of every feature map, $F_2(\boldsymbol{Y})$. GAP reduces a 3-dimensional matrix to a 1-dimensional vector. The outputs of the first fully-connected layer are calculated as follows:

$$F_5(\boldsymbol{Y}) = \max(0, W_5 \cdot F_4(\boldsymbol{Y}) + B_5), \tag{4.5}$$

where $F_5(\boldsymbol{Y})$ is an n_5-length vector. W_5 is a 2-dimensional matrix, and its size is $n_2 \times n_5$. The outputs of the second fully-connected layer are given as follows:

$$F_6(\boldsymbol{Y}) = W_6 \cdot F_5(\boldsymbol{Y}) + B_6, \tag{4.6}$$

where $F_6(\boldsymbol{Y})$ is a D-length vector, and D is the number of image categories. Let $F_{6.1}(\boldsymbol{Y})$, $F_{6.2}(\boldsymbol{Y})$, $\cdots$ separate units of $F_6(\boldsymbol{Y})$. The ith output of SoftMax function is calculated as follows:

$$F_{\mathrm{s}.i}(\boldsymbol{Y}) = \frac{\exp(F_{6.i}(\boldsymbol{Y}))}{\sum_{j=1}^{D} \exp(F_{6.j}(\boldsymbol{Y}))}, \tag{4.7}$$

where $F_{\mathrm{s}.i}$ means ith unit of F_{s}, and i is from 1 to D. Each unit of F_{s} stores the probability of the input image category.

4.2.2 Training Method

This section presents details of the training method of the proposed CNN architecture. MCH-MC has two training phases for the internal classifying system, as shown in Figures 4.5-(a) and (b). First, the CNN is pre-trained to classify input image categories in Phase1. Next, the CNN is trained to improve image quality performance for super-resolution in Phase2.

MCH-MC has two outputs: $F(\boldsymbol{Y})$ and $F_{\mathrm{s}}(\boldsymbol{Y})$; therefore, MCH-MC is trained with MTL [40]. MCH-MC requires two teaching signals of $\boldsymbol{T}$ and $\boldsymbol{t}$. $\boldsymbol{T}$ is an original high-resolution image corresponding to a low-resolution image $\boldsymbol{Y}$. The MSE loss function L_{MSE} is calculated using Equation 2.5. In contrast, $\boldsymbol{t}$ is a D-length vector with accurate category labels. The loss function is calculated with SoftMax cross entropy as follows:

$$L_{\mathrm{SCE}}(\Theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{D} \boldsymbol{t}_{ij} \log F_{\mathrm{s}.j}(\boldsymbol{Y}_i; \Theta), \tag{4.8}$$

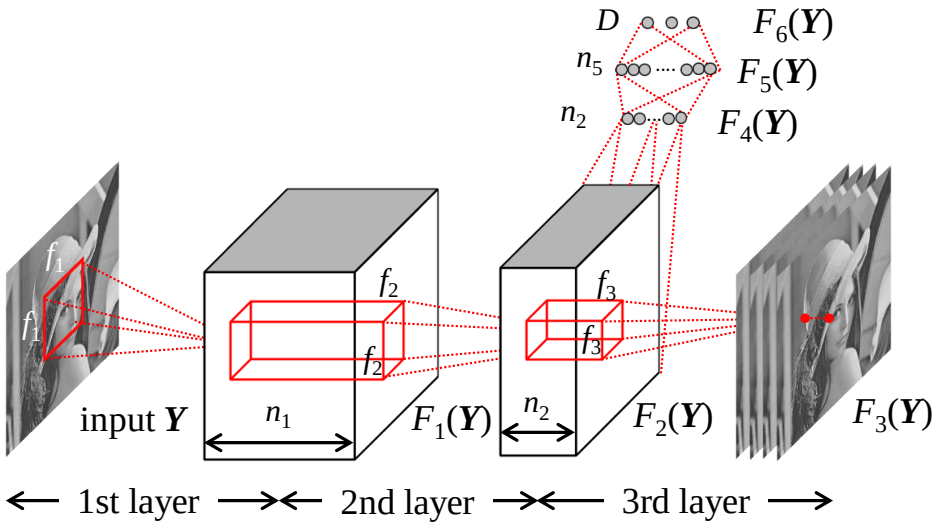


Figure 4.4: The architecture of MCH-MC.

where N is the number of training data samples, and Θ is parameters to be determined by backpropagation technique such as W_1, W_2, W_5, W_6, B_1, B_2, B_5, and B_6.

After calculating each loss, the total loss $L(\Theta)$ is defined as follows:

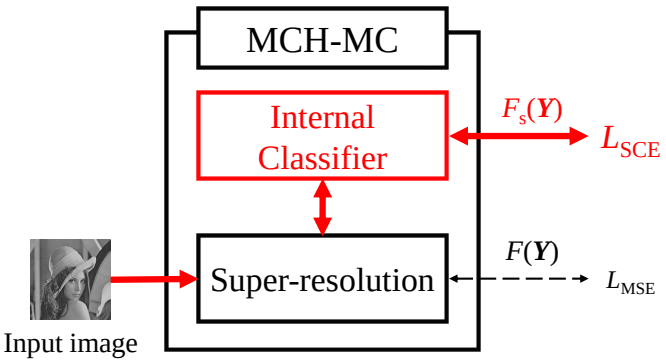
$$L(\Theta) = r_{\mathrm{MSE}} L_{\mathrm{MSE}}(\Theta) + r_{\mathrm{SCE}} L_{\mathrm{SCE}}(\Theta), \tag{4.9}$$

where r_{MSE} and r_{SCE} are composite ratios. This training method is a kind of MTL [40]. The balance between r_{MSE} and r_{SCE} is changed, as presented in Table 4.2. This section explains the details of each training phase.

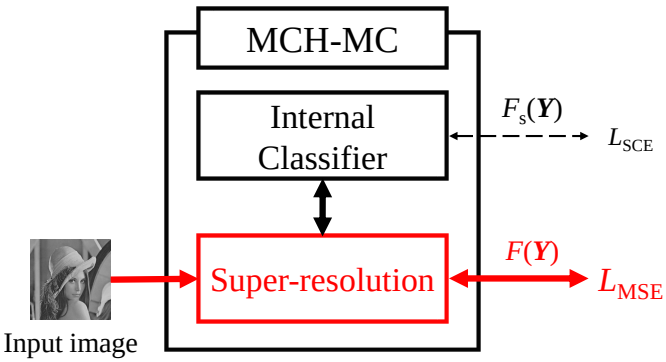
Phase1: Pre-training as Classifier

In Phase1, the CNN is pre-trained to improve the image classification ability. Therefore, r_{SCE} is set ten times larger than r_{MSE}, as presented in Table 4.2.

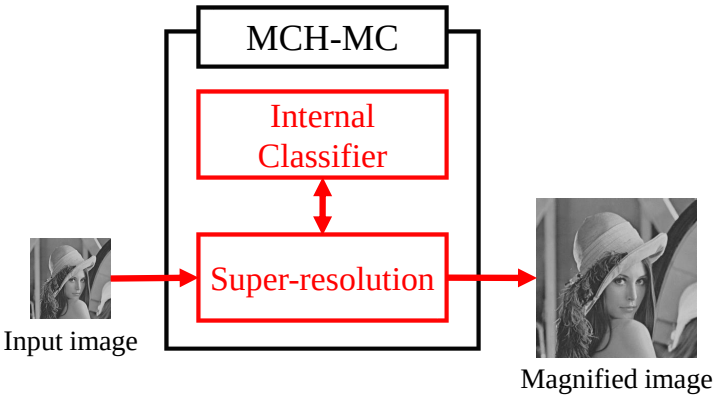
In addition, dropout [46] is inserted after the first and second convolution layers to prevent the CNN from over-optimized for image classification tasks. Dropout is a technique that randomly disables neuron of intermediate layer at the ratio D_{ratio} and improves the generalization performance by preventing overfitting. In MCH-MC, D_{ratio} is 0.5.



(a) Training Phase1 (pre-training as classifier).



(b) Training Phase2 (training for super-resolution).



(c) Inference Phase (super-resolution).

Figure 4.5: Two training phases for the internal classifying system and the inference phase.

Table 4.2: Training conditions of MCH-MC in each phase.

Phase	Loss ratio		Learning rate		Dropout ratio
	r_{MSE}	r_{SCE}	Other layers	Third convolution layer	
Phase1	0.1	1	10^{-3}	10^{-4}	0.5
Phase2	1	0.1	10^{-4}	10^{-5}	None

Phase2: Training for Super-Resolution

In Phase 2, the CNN is trained for the super-resolution. All weighted layers are inherited from the pre-trained model obtained from Phase1 [47]. Thereafter, r_{MSE} and r_{SCE} were set to 1 and 0.1 as presented in Table 4.2, respectively. Herein, $L(\Theta)$ operates to improve the super-resolution ability. r_{SCE} is set to a small value for maintaining image classification ability. Dropout layers used in Phase1 are removed from the CNN architecture. The learning rate of Phase2 are reduced from Phase1 and is set to the same values as used in Reference [14] for a fair comparison.

Approach for Unbalanced Training Dataset

If the number of training images between each category is not always the same. Then, SoftMax cross entropy is significantly influenced by the principal category that has more images than others. Consequently, although the CNN operates with higher classification accuracies for the principal category, it has lower accuracies for the remaining category. To address this problem, the MCH-MC adopts weighted SoftMax cross entropy to correct the unbalance of data samples between each category. The weighted SoftMax cross entropy is given as follows:

$$L_{\mathrm{WSCE}}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}\boldsymbol{\alpha}_j \boldsymbol{t}_{ij}\log F_{\mathrm{s}.j}(\boldsymbol{Y}_i;\Theta), \tag{4.10}$$

where the weight $\boldsymbol{\alpha}_j$ is an inversely proportional value to the quantity of data samples of category j. Consequently, the CNN can be trained appropriately, if the training dataset is nonuniform for each category. In Section 4.4, MCH-MC adopts L_{WSCE} alternate to L_{SCE}.

4.2.3 Inference Phase for Super-Resolution

After training the CNN, super-resolution processing is performed with the trained model. In this phase, the GAP and fully-connected layers used for image classification are removed, as shown in Figure 4.5-(c). Consequently, the inference CNN is the same architecture as MCH as shown in Figure 2.3. Therefore, there are no differences in the processing time and the number of parameters to be stored in the device.

However, the MCH-MC has an image classification ability in the first and second convolution layers. Section 4.3 will show this ability in later experiments. Therefore, MCH-MC can respond to the image category and change the internal behavior automatically. The users do not require specifying the input image category or determine the kind of images the CNN learned in advance. This is an advantage of the proposed method.

4.3 Evaluation Experiments

This section shows the effectiveness of the proposed method. The remainder of this section is organized as follows: Section 4.3.1 and 4.3.2 explains the experimental conditions and descriptions, respectively. Section 4.3.3 evaluates the effectiveness of the proposed method from five viewpoints. First, it evaluates the image qualities and processing times quantitatively. Second, it shows the transition in image quality performance and backpropagations. Third, it shows transition in classification accuracy and image quality. Forth, it visualizes the importance areas for the classification judgement with Grad-CAM. Finally, it shows the correlation between image quality and classification results.

4.3.1 Experimental Conditions

The experimental environment is as follows—OS: Ubuntu 16.04 LTS, CPU: Intel Core i7-8700 3.20GHz, Memory: 16.00GB, GPU: NVIDIA GeForce GTX 1080 8GB, and IDE: MATLAB.

All CNNs are implemented with a Caffe package [22]. All experiments focus only on the luminance channel in the YCrCb space because several studies of super-resolution are evaluated only on the luminance channel [3, 10, 11, 13, 14, 23–27]. This experiment evaluates 2× image magnification. PSNR and SSIM are used for objective evaluation and calculated with the same as Equation 2.15 and Equation 2.16, respectively. PSNR and SSIM are calculated only at the

Table 4.3: Training and testing datasets.

Dataset	Amount	The average number of pixels	Category name
BSDS100 [29]	100	154,401	"Nature"
Manga109 [35, 36]	109	966,011	"Manga"
Hradiš's Dataset [39]	100	40,000	"Text"

Table 4.4: Parameter setting of CNN for 2× magnification.

MCH·MCH-MC	MCH-MC
$\{f_1, f_2, f_3\}$={5, 5, 3}	n_5=256
$\{n_1, n_2\}$={64, 32}	D=3

Table 4.5: Setting of weighted SoftMax cross entropy parameter, "$\boldsymbol{\alpha}$," on MCH-MC.

K-fold	$\boldsymbol{\alpha}_{Nature}$	$\boldsymbol{\alpha}_{Manga}$	$\boldsymbol{\alpha}_{Text}$
1	0.4760	0.0888	1
2	0.4760	0.0888	1
3	0.4760	0.0888	1
4	0.4751	0.0886	1
5	0.4995	0.0931	1

center of the luminance channel to prevent the influence of the image boundaries.

4.3.2 Experimental Descriptions

This experiment evaluates the performances by K-fold cross-validation. The number of divisions is five.

The training and testing images are three datasets, as presented in Table 4.3. This experiment defines BSDS100, which is a dataset of natural images, Manga109, which is a dataset of comic images, and Hradiš's dataset, which is a dataset of text images, as "Nature," "Manga," and "Text," respectively.

The training phase prepares 60×60 pixel teaching signals, $\boldsymbol{T}$, cropped from original images with a stride s pixel. The sizes of stride s are five for "Nature" and "Text," and 10 for "Manga."

Table 4.6: Training datasets and the CNN architecture of each method.

Method	Training dataset			Architecture
	Nature	Manga	Text	
SRforNature	✓			MCH
SRforManga		✓		MCH
SRforText			✓	MCH
SRforAll	✓	✓	✓	MCH
SRforMC	✓	✓	✓	MCH-MC

The input signals, $\boldsymbol{Y}$, are prepared by downsampling these teaching signals with Bicubic interpolation of 1/2 scaling factor. The MSE loss function is optimized with the central pixels of $\boldsymbol{T}$ because all convolution layers have no padding. The teaching signals of image category $\boldsymbol{t}$ are vectors of three elements corresponding to "Nature," "Manga," and "Text." The training phase do not use any data augmentation for the training and testing images. According to [14], the filter sizes of CNN are as shown in Table 4.4. The filter weights of each layer are initialized by random values from a Gaussian distribution with a mean and standard deviation of 0 and 0.001, respectively, and 0 for biases. The Adam optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [48]. The aforementioned parameters are common for all the methods. This experiment set the weight, $\boldsymbol{\alpha}$, of the MCH-MC as presented in Table 4.5. The number of backpropagations of Phase1 in the MCH-MC is 1 million (32 batch size). The number of backpropagations of the MCH and MCH-MC in Phase2 are 10 million (32 batch size) at the maximum. This experiment examines the following five methods:

- SRforNature
- SRforManga
- SRforText
- SRforAll
- SRforMC (SRforMulti-Category)

These methods have different CNN architectures and training images as shown in Table 4.6.

Table 4.7: PSNRs and SSIMs of 2× magnified images.

Method	PSNR[dB] / SSIM			
	Nature	Manga	Text	Average
SRforNature	**31.31** / **0.8883**	35.48 / 0.9656	23.25 / 0.9203	30.17 / 0.9259
SRforManga	31.20 / 0.8868	**36.65** / **0.9698**	24.53 / 0.9423	30.96 / 0.9340
SRforText	28.08 / 0.7772	29.77 / 0.8711	24.66 / 0.9426	27.57 / 0.8639
SRforAll	31.22 / 0.8870	36.25 / 0.9683	25.28 / 0.9510	31.07 / 0.9364
SRforMC	**31.31** / **0.8883**	36.46 / 0.9694	**25.57** / **0.9565**	**31.29** / **0.9389**

Bold red and underlined blue fonts indicate the best and second-best scores, respectively.

4.3.3 Experimental Result and Discussion

Performance of Image Qualities and Processing Times

Table 4.7 lists the PSNRs and SSIMs for 2× magnified images of testing datasets. Bold red and underlined blue fonts indicate the highest and second highest values, respectively. First, this section focuses on each category. On category "Nature," SRforNature and SRforMC have almost the same image quality performances. On category "Manga," SRforManga achieved the highest image quality performances. On category "Text," however, SRforText did not achieve the highest image quality performance, because the number of training images used for SRforText is significantly fewer than that for other methods. From these results, CNN-based super-resolution achieves a good image quality if sufficient number of images of the same category is learned previously. However, if there are few training images of the same category, the average PSNR and SSIM are degraded.

Next, this section focuses on SRforAll and SRforMC. These methods have the same training datasets; however, they have different CNN architecture. SRforMC achieved a 0.22pt higher average PSNR and a 0.0025pt higher average SSIM than SRforAll. This is because MCH-MC is trained with a category information. Thus, SRforMC is the best solution if the input image category is variable.

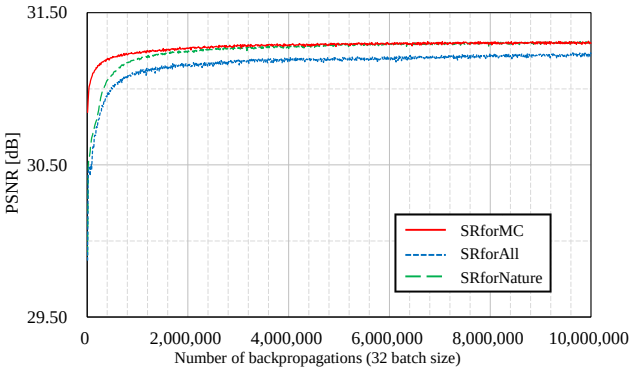
Furthermore, this section evaluates the image quality subjectively. Figure 4.6 shows examples of magnified images. There are significant subjective degradations in "108005" by SRforText, "DualJustice" by SRforText, and "0000001" by SRforNature. The reason is an inconsistency between the input and training image categories. Whereas SRforText enhances the edges, SR-

Figure 4.6: Visual comparison on 2× magnified images. Bold red and underlined blue fonts indicate the best and second-best scores in each testing image, respectively.

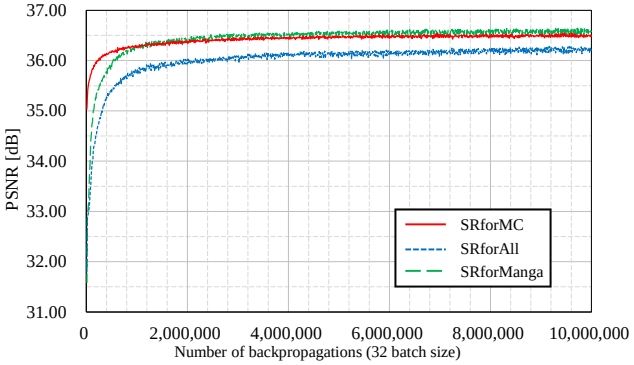
forNature smoothens them. Therefore, it is necessary to consider the training image category of the CNN before using it.

Finally, this section evaluates the processing time on a CPU. GPU acceleration is not used in this evaluation because MCH is developed for low complexity and high-speed processing [14]. Table 4.8 shows the processing time for each architecture at inference. There was no difference in processing time because the MCH-MC of the inference mode has the same CNN architecture as the MCH shown in Figure 2.3. Consequently, the processing times of all methods were the same.

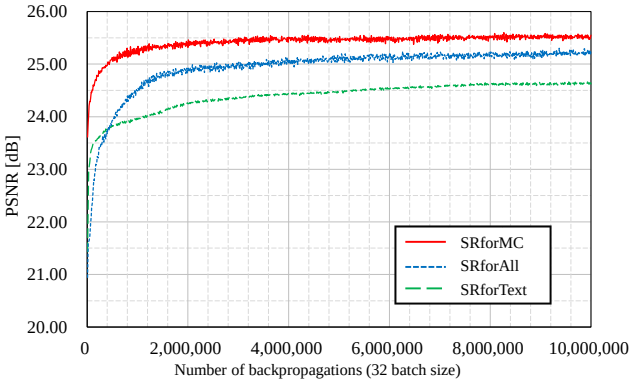
However, a comparison of super-resolutions is basically evaluated with standard benchmark training and testing datasets including T91 [28], BSDS200 [29], Set5 [30], Set14 [31], and BSDS100 [29]. The comparison of MCH and other super-resolutions is stated in [14]; therefore, this experiment do not consider it in this chapter.



(a) Nature.



(b) Manga.



(c) Text.

Figure 4.7: The average PSNR curves for the number of backpropagations (2× magnification). The horizontal and vertical axes of these curves indicate the number of backpropagation and the average PSNR, respectively.

Table 4.8: Processing time of 2× magnified images in all categories.

Processing time[s]	
MCH	MCH-MC
0.8003	0.8003

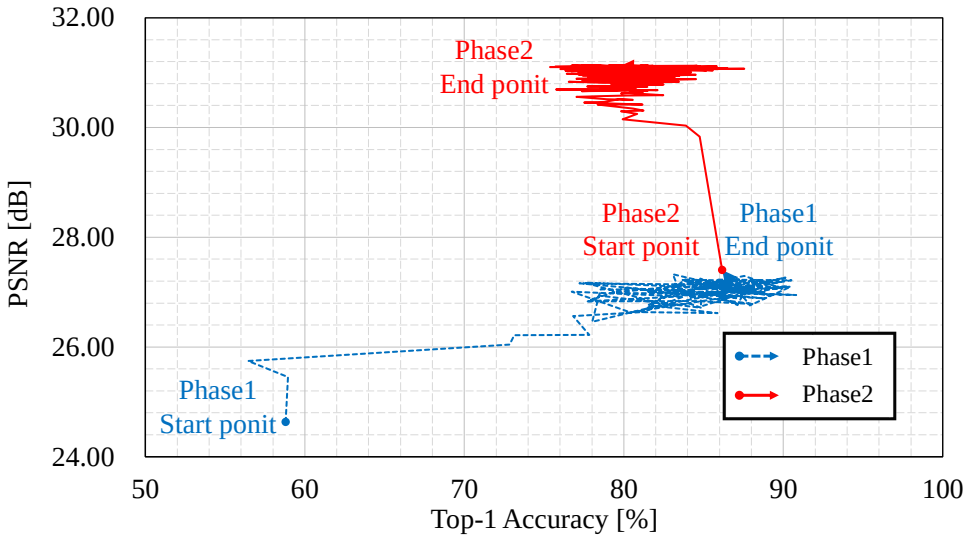


Figure 4.8: Transitions in classification accuracy and image quality on SRforMC. The horizontal and vertical axes in this figure indicate the average accuracy of image classification and the average PSNR, respectively.

PSNR Convergence Curves

Figure 4.7 shows the PSNR convergence curves. In the figure, the PSNR curves of SRforMC shows those of Phase2. In general, the image quality performance of CNN improves with the number of backpropagations. SRforMC achieved higher PSNRs than SRforAll at the same number of backpropagation points. This advantage will may not change after 10 million back-propagations.

Transition in Classification Accuracy and Image Quality

Figure 4.8 shows the transition in classification accuracy and image qualities in SRforMC for all testing images. The Phase1 start point is at the lower left of this figure and is obtained after

Table 4.9: Top-1 accuracy of image classification in SRforMC.

Top-1 Accuracy[%]			
Nature	Manga	Text	Average
72.0	96.4	89.0	85.8

1,000 backpropagations. Next, the point moves to the right to indicate that the CNN is trained to improve image classification accuracy. Subsequently, the point moves to the top, indicating that CNN is trained to improve the PSNRs while maintaining the classification accuracy.

Table 4.9 presents the final accuracies of image classification in SRforMC. The average accuracy is 85.8 %. Therefore, SRforMC has both the super-resolution and image classification abilities.

Visualization of Classification Ability

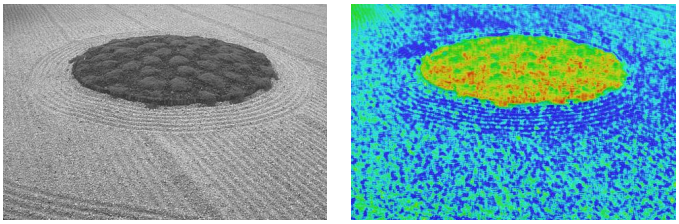
This section subjectively evaluates the CNN's internal behavior for each category with gradient-weighted class activation mapping (Grad-CAM) [49]. Grad-CAM is a technique that considers the important area for assessments as a heat map. In this experiment, Grad-CAM shows which area is recognized as "Nature," "Manga," or "Text" in the second feature map, $F_2(\boldsymbol{Y})$.

Figure 4.9 shows some examples of Grad-CAM outputs for each category. By observing the red and yellow areas, "Nature," "Manga," and "Text" are based on uneven, flat, and edge areas, respectively. Thus, the first and second convolution layers extract important characteristics for the classification.

Correlation between Image Quality and Classification

This section evaluates whether the image quality depends on the classification accuracy. Table 4.10 shows the increase in the average PSNRs from SRforAll to SRforMC according to the success or failure of the category classification. The increase in the PSNRs and SSIMs is larger in the correctly classified image group than in the misclassified image group.

In contrast, some image samples improved in image quality despite misclassification; therefore, this result will be discussed. Figure 4.10 shows the increase in PSNR from SRforAll to SRforMC for each testing image. The horizontal and vertical axes in this figure indicate the PSNR increase



Input image
(Nature: “86016”)

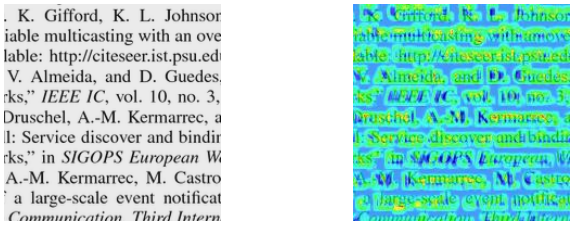
Output of Grad-CAM

(a) Important areas classified as “Nature.”

Input image
(Manga: “MariaSamaNihaNaisyo”)

Output of Grad-CAM

(b) Important areas classified as “Manga.”



Input image
(Text: “0000099”)

Output of Grad-CAM

(c) Important areas classified as “Text.”

Figure 4.9: Importance of judgment grounds in each category classification.

Table 4.10: Increase in image qualities from SRforAll to SRforMC.

Increase of PSNR[dB] / Increase of SSIM	
True	False
+0.20 / **+0.0027**	+0.14 / +0.0017

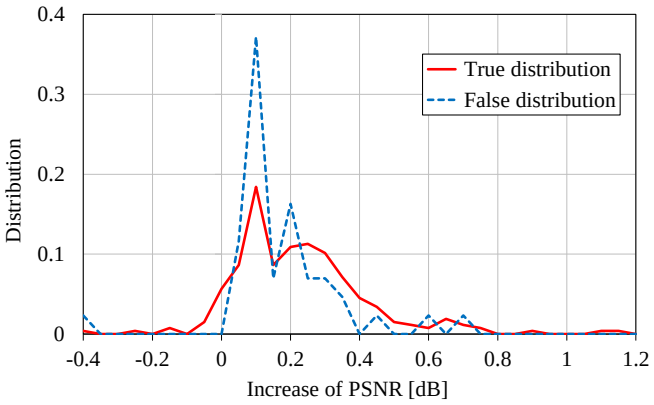


Figure 4.10: Increase in PSNR from SRforAll to SRforMC depending on the classification result. The horizontal and vertical axes in this figure indicate the PSNR increase from SRforAll to SRforMC and their distributions, respectively.

from SRforAll to SRforMC and their distributions, respectively. The distribution of correct classification spreads to the right compared to that of misclassification. This result indicates that these misclassified images insignificantly improved in image quality than the correctly classified images. However, some misclassified images achieved significant image quality improvement.

Figure 4.11 shows an example that significantly improved the PSNR despite the misclassification. This sample is a “Text” with a large font size. Then, the CNN estimated it as a “Manga” with a probability of 86.8% or a “Text” with a probability of 10.7%. Grad-CAM shows that the black flat parts caused the “Manga” probability, and the edge parts caused the “Text” probability. Because the former probability is larger than the latter, the CNN seems to estimate it as a “Manga.” However, SRforMC improved the 0.58pt PSNR and reduced the overshoots around the edge area, as shown in the enlarged images. Therefore, the proposed method made accurate decisions locally, leading to the improved image quality. Therefore, SRforMC can perform appropriate processing based on the image characteristics, although image categories are not accurately classified.

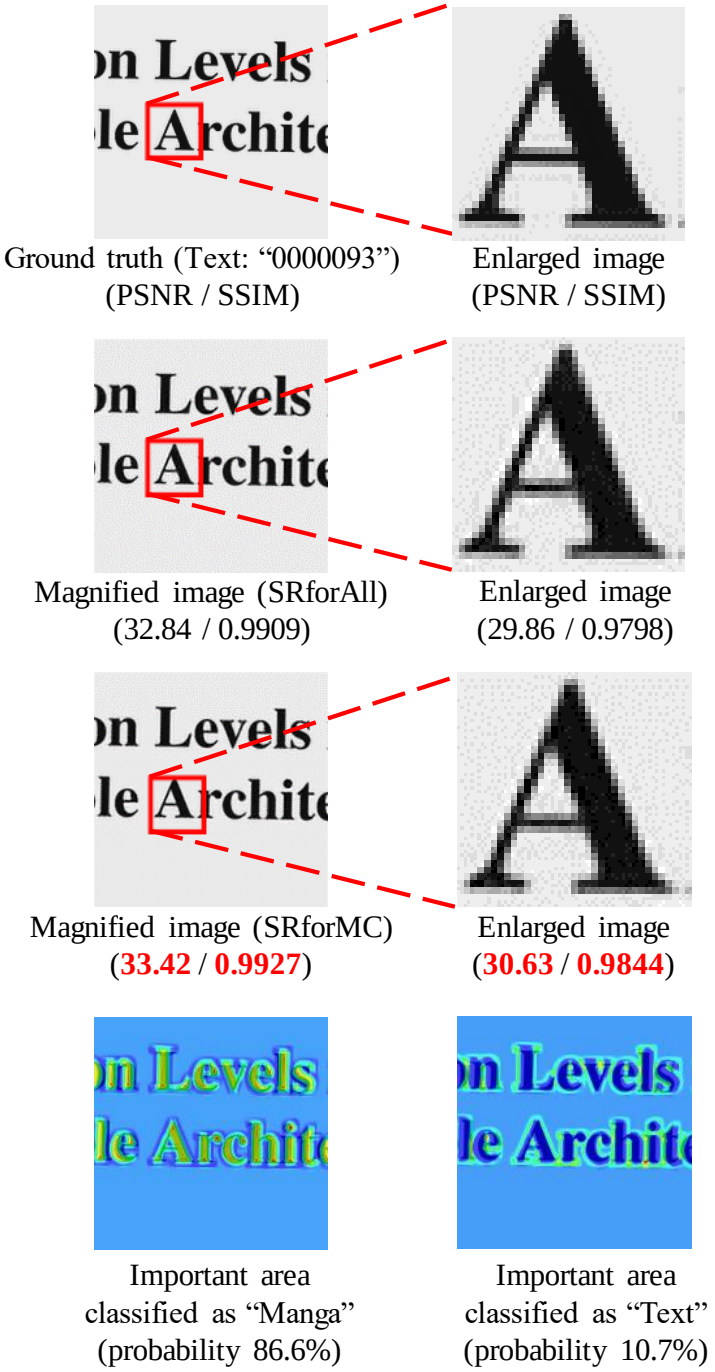


Figure 4.11: Image sample with a large quality improvement despite misclassification in SRforMC.

4.4 Conclusion

This chapter proposes an image super-resolution for multiple image categories with a single CNN and MTL. The proposed CNN does not require an external category information; however, it has an internal category-classifying ability. In experiments, the average PSNR of the proposed method was approximately 0.22 dB higher than that of the conventional method with the same training dataset and processing time. The proposed method is useful when the input image category is varying. Implementation for a single image containing multi-category will be considered in the future studies.

5 Automated Fish Bone Detection in X-ray Images with Convolutional Neural Network and Synthetic Image Generation

5.1 Introduction

Food safety issues are increasingly attracting attention. The foreign objects found in food include metals, glass, stones, and bones. The contamination occurs during production, storage, and distribution. These accidents not only decrease the value of food and impact human health but also damage the credit of the company. To prevent contamination by foreign objects, X-ray detectors are used during processing lines. Imaging techniques have advantages in the non-destructive detection of foreign objects [50].

This chapter focuses on fish bone, which is one of the most difficult foreign objects to detect. The X-ray detector can visualize bones in fish using the same principle as radiography. Currently, workers watch the X-ray images, and when they detect a fish bone, they remove it. However, the visual inspection by humans has two problems. One is difficulty in maintaining the detection accuracy because it depends on the experience of the workers. The other is human error due to fatigue. Given these, the development of automated fish bone detection technique is needed.

Mery et al. detected bones in fish fillets [51]. First, the X-ray images were divided into small blocks. Next, features were extracted from each block. Finally, a support vector machine classified them into bone or non-bone blocks. However, this method required manual annotations of the images for the training phase.

Meanwhile, semantic segmentation techniques with CNNs have achieved state-of-the-art results in pixel-wise object detection [52–54]. These methods detect a specific object in pixel units and output their position and shape. Long et al. proposed fully convolutional neural networks for semantic segmentation (FCN) [52]. They replaced fully connected layers with convolution layers in popular image classification CNNs, such as Alexnet [1] and VGG [55]. These architectures output pixel-wise probabilities of input image classes. Ronneberger et al. proposed the

U-Net with encoder-decoder architecture which is a type of FCN [53]. This architecture was proposed for biomedical image semantic segmentation.

The difficulty in implementing semantic segmentation CNNs is collecting the training dataset. CNNs with supervised learning generally need a large number of training images and teaching signals to achieve good performance. If a large number of data samples are already prepared in advance, they can be used to train the CNN. For example, image datasets of PASCAL VOC [56] and COCO [57] provide many training images and their pixel-wise teaching signals. However, there are many areas where only a limited number of data samples are prepared, or a sufficient number of data samples cannot be prepared without a significant cost. Moreover, two problems are encountered if they are used for fish bone detection:

1. Manual annotations of X-ray images are difficult because fish bones are very thin.

2. It is difficult to sample all variations of fish bones with various lengths, angles, and thicknesses.

This chapter proposes a synthetic image generation technique using computer graphics for a semantic segmentation CNN to overcome above problems. The proposed method generates fish bone-like silhouettes and draws them on X-ray images. Because the training images and their pixel-wise teaching signals are generated simultaneously, it is not needed to manually annotate the dataset.

5.2 Fish Bone Detection with Convolutional Neural Network and Synthetic Image Generation

This section presents a new fish bone detection technique using a CNN and synthetic image generation. Figure 5.1 shows the X-ray detector of Ishida Co., Ltd [58]. This detector projects X-rays from the top light source, and the lower line sensors image the fish. Figure 5.2-(a) shows an example of an X-ray image of a fish. Fish bones shown in Figure 5.2-(b) are difficult to discern; however, they exist. Finding such small bones and preparing training data are challenging. Therefore, this section proposes to simulate these bones and create a virtual dataset.

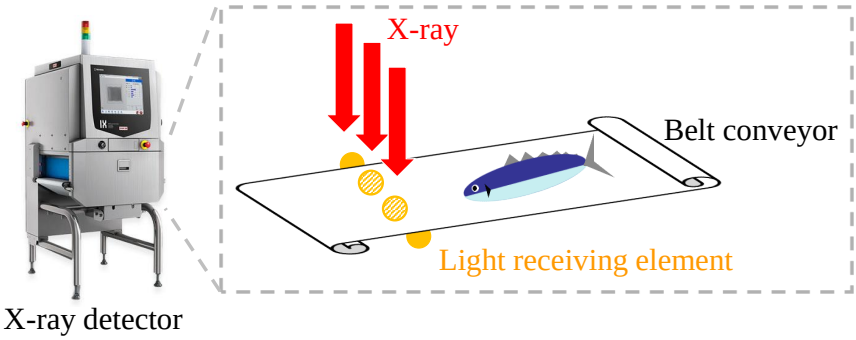
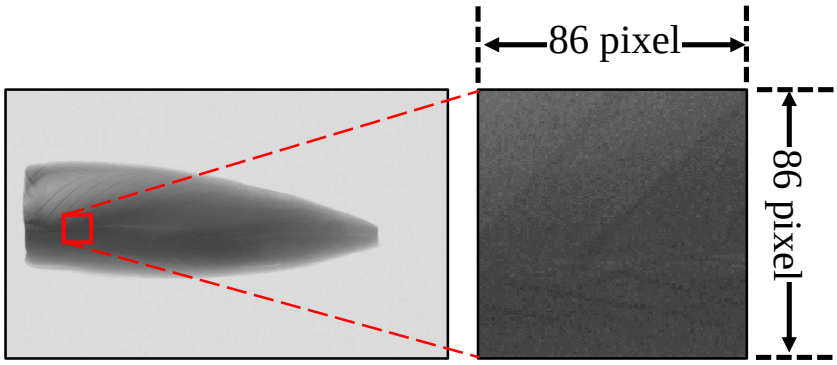
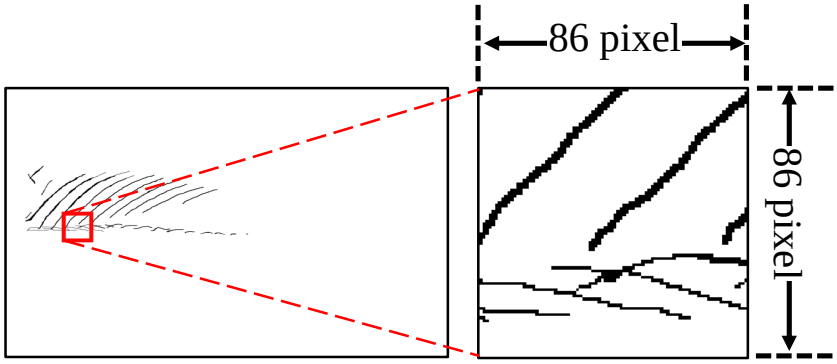


Figure 5.1: X-ray imaging with line sensor.



(a) X-ray image of fish.



(b) bone area.

Figure 5.2: X-ray image of fish bone and its bone area. In these enlarged figures, the darkness and thickness of the bones range approximately from 3 to 20 and from 1 to 5 pixel, respectively.

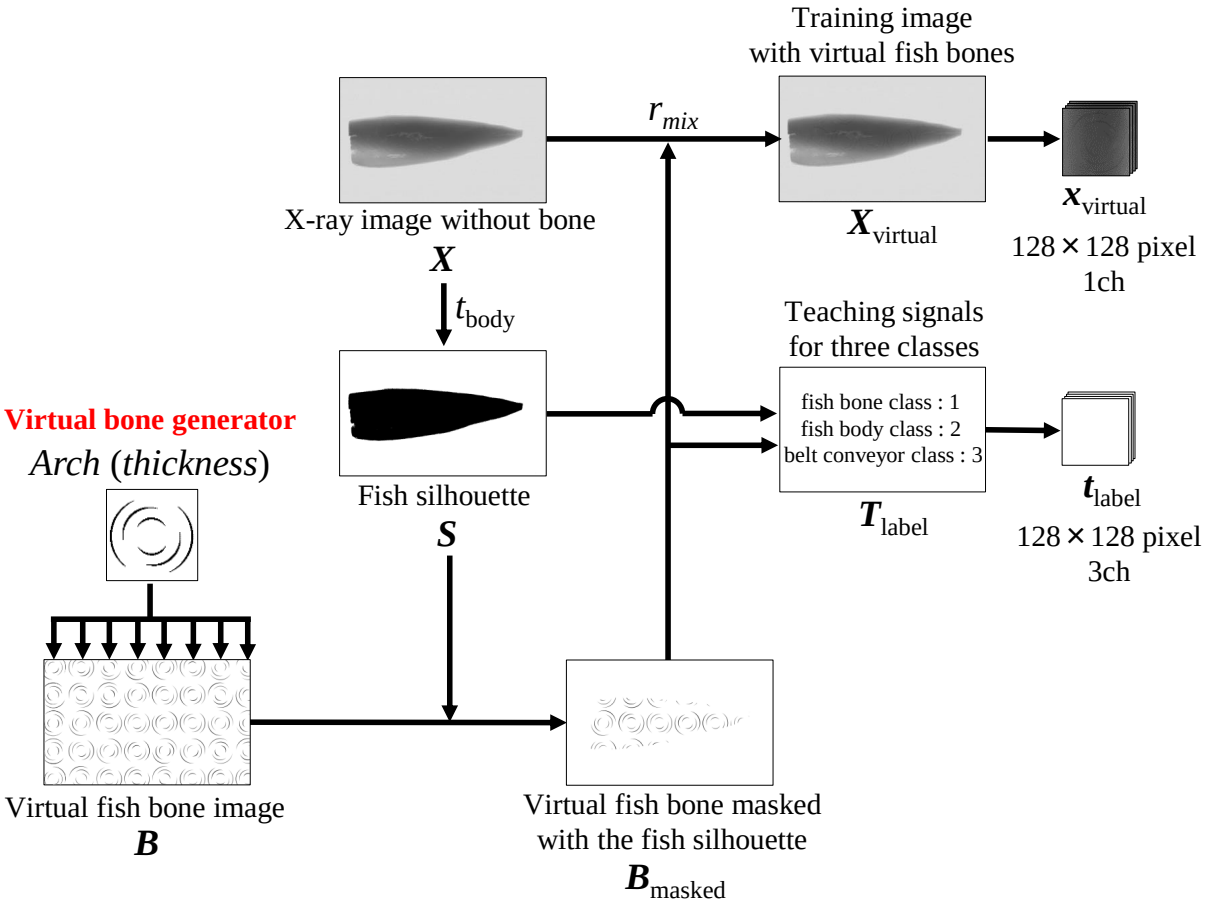


Figure 5.3: Process of the proposed synthetic image generation.

5.2.1 Synthetic Image Generation with Virtual Fish Bones

Figure 5.3 shows the proposed synthetic image generation. The details are described in the following four subsections.

Separation of Fish Body and Belt Conveyor Areas

The first step in Figure 5.3 acquires the X-ray image $\boldsymbol{X}$ that does not include fish bones. In $\boldsymbol{X}$, the luminance values of the fish body area are smaller than those of the belt conveyor area. Using this property, the fish body silhouette $\boldsymbol{S}$ is extracted. The binary fish silhouette image is

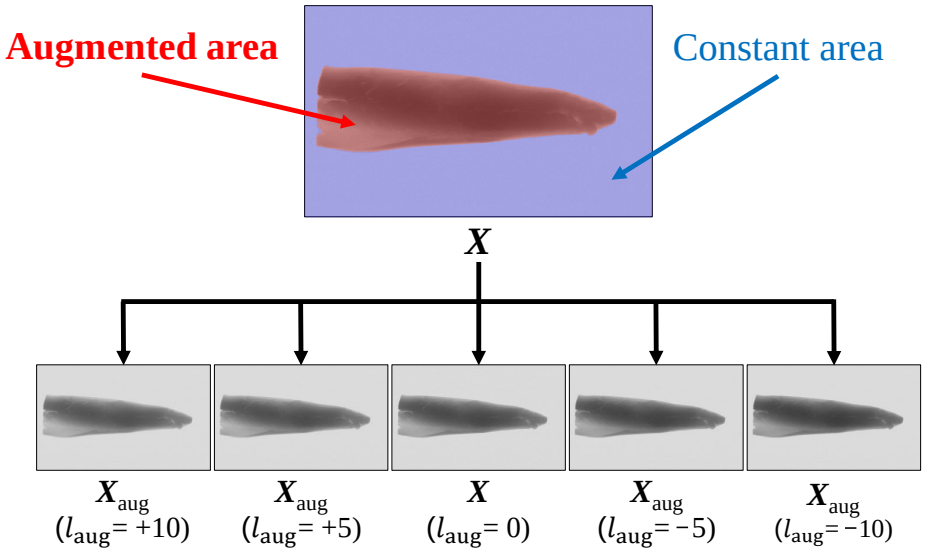


Figure 5.4: Data augmentation for an image of fish body.

obtained as follows:

$$\boldsymbol{S}(x,y) = \begin{cases} 0 & \text{if } \boldsymbol{X}(x,y) \leq t_{\text{body}} \\ 255 & \text{otherwise} \end{cases}, \tag{5.1}$$

where (x, y) and t_{body} are the pixel positions in the image and the threshold of the luminance value for image segmentation, respectively. t_{body} is set to separate the fish body and belt conveyor areas.

Data Augmentation for Fish Body

Data augmentation is one of the most widely used techniques for boosting performance with deep learning [59]. This technique increases the variations of images by transforming the original data several times.

The amount of X-ray transmission depends on the thickness of the fish body. To simulate many variations of the thickness, as shown in Figure 5.4, the proposed method transforms the luminance values of the fish body area on $\boldsymbol{X}$. In contrast, the luminance value of the belt conveyor area is always constant. Thus, the augmented image $\boldsymbol{X}_{\text{aug}}$ is generated as follows:

$$\boldsymbol{X}_{\text{aug}}(x,y) = \begin{cases} \boldsymbol{X}(x,y) + l_{\text{aug}} & \text{if } \boldsymbol{S}(x,y) = 0 \\ \boldsymbol{X}(x,y) & \text{otherwise} \end{cases}, \tag{5.2}$$

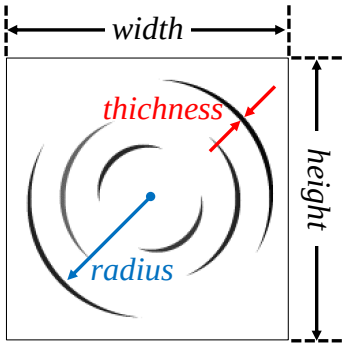


Figure 5.5: Sizes of *Arch*. In this chapter, *thickness* is set to 5, *height* and *width* are set to 200, and *radius* is set randomly. The darkness and rotation angle of this pattern change randomly for each drawing because actual fish bones have various curvatures.

where l_{aug} is the offset of the luminance value and is set to ± 5 or ± 10. As a result, the number of $\boldsymbol{X}$ is augmented five times.

Generation of Virtual Fish Bones

The function *Arch* generates a fish bone pattern, as shown in Figure 5.5. This pattern is designed for including various curvatures, angles, and thicknesses. Although Figure 5.5 is not enough to cover all the actual bones, this chapter considers it one of the effective designs. As shown in Figure 5.3, this pattern is drawn repeatedly on $\boldsymbol{B}$, where $\boldsymbol{B}$ is a white image of the same size as $\boldsymbol{X}$. Next, the drawing area of the virtual fish bone $\boldsymbol{B}$ is limited by the fish silhouette $\boldsymbol{S}$ as follows:

$$\boldsymbol{B}_{\text{masked}}(x,y) = \begin{cases} \boldsymbol{B}(x,y) & \text{if} \quad \boldsymbol{S}(x,y) = 0 \\ 255 & \text{otherwise} \end{cases}. \tag{5.3}$$

Thus, the virtual fish bones are drawn only on the fish body.

Generating Training Images and Teaching Signals

Using the X-ray $\boldsymbol{X}$ and virtual fish bone images $\boldsymbol{B}_{\text{masked}}$, the training image $\boldsymbol{X}_{\text{virtual}}$ is generated as follows:

$$\boldsymbol{X}_{\text{virtual}}(x,y) = \boldsymbol{X}(x,y) - r_{\text{mix}}(255 - \boldsymbol{B}_{\text{masked}}(x,y)), \tag{5.4}$$

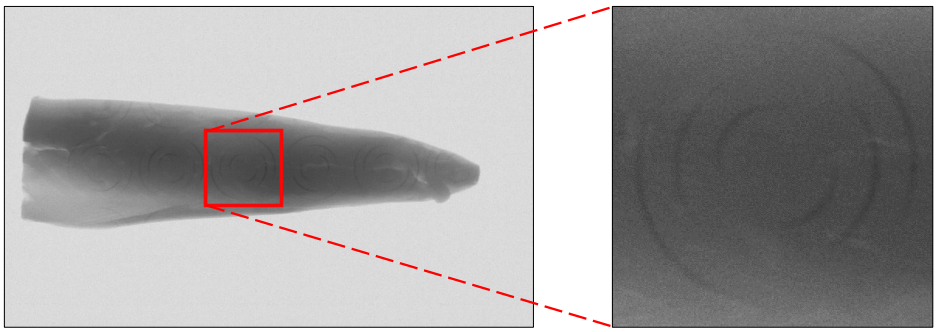


Figure 5.6: An example of training image $\boldsymbol{X}_{\text{virtual}}$. Fish bone-like silhouettes are drawn on the fish body.

where r_{mix} is the synthetic ratio of $\boldsymbol{B}_{\text{masked}}$ to $\boldsymbol{X}$. In the experiments, r_{mix} is subjectively set to 0.05 so that the darkness of the virtual bones is similar to that of the actual ones. This value should be adjusted according to the fish type and the specification of X-ray detector. Figure 5.6 shows an example of $\boldsymbol{X}_{\text{virtual}}$. Finally, the teaching signal $\boldsymbol{T}_{\text{label}}$ is generated as follows:

$$\boldsymbol{T}_{\text{label}}(x,y) = \begin{cases} 1 : \text{fish bone class} & \text{if } \boldsymbol{B}_{\text{masked}}(x,y) \neq 255 \\ 2 : \text{fish body class} & \text{else if } \boldsymbol{S}(x,y) = 0 \\ 3 : \text{belt conveyor class} & \text{otherwise} \end{cases} . \tag{5.5}$$

The teaching signal $\boldsymbol{T}_{\text{label}}$ has a three-class label for each pixel. Incidentally, if the size of $\boldsymbol{X}_{\text{virtual}}$ is larger than 128×128 pixels, the final CNN training image $\boldsymbol{x}_{\text{virtual}}$ and its teaching signal $\boldsymbol{t}_{\text{label}}$ are cropped to 128×128 pixels from $\boldsymbol{X}_{\text{virtual}}$ and $\boldsymbol{T}_{\text{label}}$, respectively.

5.2.2 Convolutional Neural Network for Fish Bone Detection

Proposed CNN Architecture

As shown in Figure 5.7, the proposed CNN architecture is a customized U-Net, which was developed for fast and accurate semantic segmentation [53]. Compared with the original study [53], proposed U-Net adds batch normalization [20] before the ReLU activation function [18] and uses dropout [46] after ReLU in its encoder. The dropout rate r_{drop} is set to 0.4.

The input signal of the CNN is a grayscale image. The proposed CNN outputs the pixel-wise object detection results of three channels. The ith channel output of the SoftMax function is

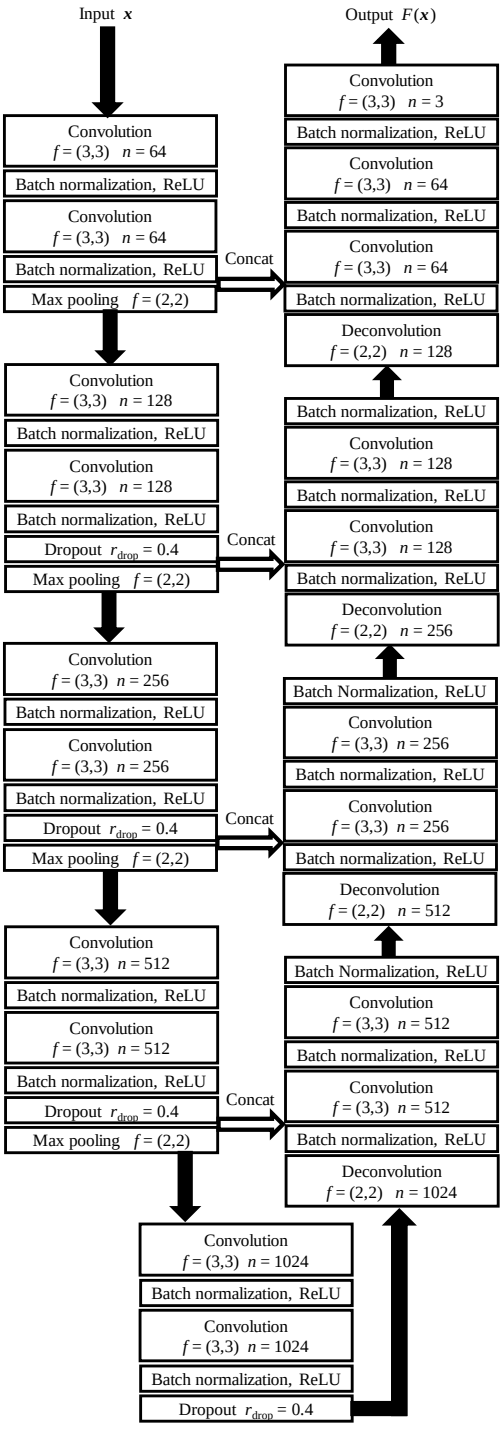


Figure 5.7: The proposed CNN architecture based on U-Net. f, n, and r_{drop} denote the filter size, the number of filters, and the ratio of dropping nodes, respectively.

calculated as follows:

$$F_{\mathrm{s}.i}(\boldsymbol{x}_{\mathrm{virtual}}) = \frac{\exp(F_i(\boldsymbol{x}_{\mathrm{virtual}}))}{\sum_{j=1}^{3}\exp(F_j(\boldsymbol{x}_{\mathrm{virtual}}))}, \tag{5.6}$$

where $F_i(\boldsymbol{x}_{\mathrm{virtual}})$ and $F_j(\boldsymbol{x}_{\mathrm{virtual}})$ are the ith and jth channels of the outputs from the proposed CNN, respectively. Every channel $F_{\mathrm{s}.i}(\boldsymbol{x}_{\mathrm{virtual}})$ stores the pixel-wise probability of every class.

In the test phase, a test X-ray image $\boldsymbol{X}_{\mathrm{test}}$, which includes actual fish bones, is sent to the trained CNN directly. The trained CNN outputs the pixel-wise detection result. The binary output image $F_{\mathrm{bone}}(\boldsymbol{X}_{\mathrm{test}})$ is generated as follows:

$$\begin{aligned} &F_{\mathrm{bone}}(\boldsymbol{X}_{\mathrm{test}})(x,y) \\ &\quad = \begin{cases} 0 & \text{if} \quad \underset{\{i=1,2,3\}}{\arg\max}\, F_{\mathrm{s}.i}(\boldsymbol{X}_{\mathrm{test}})(x,y) = 1 \\ 255 & \text{otherwise} \end{cases}. \end{aligned} \tag{5.7}$$

Training Method

The semantic segmentation CNN is generally optimized with pixel-wise SoftMax cross entropy [52–54]. However, in X-ray images, the pixel values of every class are significantly different from each other. In this situation, the SoftMax cross entropy is greatly influenced by the major class, which has more pixels than the others. Thus, the proposed CNN is optimized with pixel-wise weighted SoftMax cross entropy (WSCE) as follows:

$$L_{\mathrm{WSCE}}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}\boldsymbol{\alpha}_j \boldsymbol{t}_{\mathrm{label}\cdot ij}\log F_{\mathrm{s}.j}(\boldsymbol{x}_{\mathrm{virtual}\cdot i};\Theta), \tag{5.8}$$

where N, $\boldsymbol{\alpha}_j$, and Θ are the number of training samples, the weight factor in class j, and the parameters to be determined by the backpropagation technique, respectively. In Equation. 5.8, $\boldsymbol{t}_{\mathrm{label}}$ is encoded as a one-hot vector in advance. $\boldsymbol{\alpha}_j$ is inversely proportional to the number of pixels in class j.

5.3 Evaluation Experiments

This section evaluates the performances of the two methods shown in Table 5.1. The CNN-AFB (actual fish bone) learned actual fish bones; however, the number of training images was only 10 because manual annotation is difficult. On the other hand, the CNN-VFB (virtual fish

Table 5.1: Experimental conditions.

Method	Training images	Evaluation
CNN-AFB	actual fish bones ×10 images	five-fold cross validation
CNN-VFB	virtual fish bones ×120 images	non-cross validation

bone) learned only virtual fish bones utilizing 120 training images, which were automatically generated using the proposed method.

The target fish type was mackerel in this experiment. All 896×1480 pixels X-ray images were obtained from the X-ray detector of Ishida Co., Ltd. First, 12 X-ray images $\boldsymbol{X}$ without fish bones were obtained for synthetic image generation. $\boldsymbol{X}$ was augmented five times and $\boldsymbol{B}$ was generated twice. Therefore, the total number of training images $\boldsymbol{X}_{\text{virtual}}$ was 120. Next, 10 X-ray images $\boldsymbol{X}_{\text{test}}$ with fish bones were obtained for test. Note that $\boldsymbol{X}_{\text{test}}$ includes actual fish bones. The ground truth images of $\boldsymbol{X}_{\text{test}}$ were produced by professionals of fish bone detection.

In the evaluation of the CNN-AFB, cross validation was used because there were only 10 images with actual fish bones. They were divided into five folds. Four folds are used as training data, and the remaining fold was used as test data.

The training conditions of both CNN-AFB and CNN-VFB were as follows: The number of backpropagations was 500,000 (8 batch sizes) at the maximum. The optimizer used was Adam [48]. The initial learning rate was set to 1.0×10^{-4} and reduced by 0.5 at 250,000, 375,000, 437,500, and 468,750 backpropagations. The filter weights of every convolution layer were initialized using the Xavier algorithm [60]. CNNs were implemented with a Caffe package [22].

5.3.1 Objective Evaluation of Detection Performance

First, this section evaluates pixel-wise detection accuracy with precision, recall, F-measure, and intersection over union (IoU) [54]. Let N_{TP}, N_{FP}, N_{TN}, and N_{FN} be the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) pixels in the fish bone detection, respectively. These values are calculated between the binary ground truth image and the binary output of CNN $F_{\text{bone}}(\boldsymbol{X}_{\text{test}})$. Precision P indicates the accuracy of output and

is calculated as follows:

$$P = \frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}} + N_{\mathrm{FP}}}. \tag{5.9}$$

Recall R indicates the completeness of output and is calculated as follows:

$$R = \frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}} + N_{\mathrm{FN}}}. \tag{5.10}$$

F-measure indicates the harmonic average of precision and recall. It is calculated as follows:

$$\text{F-measure} = \frac{2PR}{P + R}. \tag{5.11}$$

IoU indicates how accurately the output matches the ground truth. It is calculated as follows:

$$\mathrm{IoU} = \frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}} + N_{\mathrm{FP}} + N_{\mathrm{FN}}}. \tag{5.12}$$

Table 5.2 shows the objective evaluation results. The CNN-VFB achieved a moderately good average precision of 0.738; however, this value is lower than that of CNN-AFB. This is because CNN-VFB is not trained with actual fish bones of mackerel. It implies that the virtual fish bones are moderately different from the actual ones. It is needed to improve the reality of proposed virtual fish bones. By contrast, the average recall of CNN-VFB was 0.760, which is considerably higher than that of CNN-AFB. This is because the CNN-VFB can learn various variations of bone shapes with the synthetic image generation, while the CNN-AFB cannot find unknown patterns that are not included in the training data. Consequently, the average F-measure of CNN-VFB was 0.747, which is 0.254 point higher than that of CNN-AFB.

5.3.2 Trade-off between Precision and Recall

Next, this section discusses the trade-off between precision and recall with a precision-recall curve (PR curve). The PR curve provides the performance of a binary decision problem. Figure 5.8 shows the PR curves drawn by changing the threshold t_{fish} as follows:

$$F_{\mathrm{bone}}(\boldsymbol{X}_{\mathrm{test}})(x, y) = \begin{cases} 0 & \text{if } \ F_{\mathrm{s.1}}(\boldsymbol{X}_{\mathrm{test}})(x, y) \geq t_{\mathrm{fish}} \\ 255 & \text{otherwise} \end{cases}. \tag{5.13}$$

Figure 5.9 shows an output of fish bone class $F_{\mathrm{s.1}}(\boldsymbol{X}_{\mathrm{test}})$ as a heat map in CNN-VFB. In Figure 5.8, the area of the under the precision-recall curve (AUC-PR) is calculated by the composite trapezoidal method [61] as follows:

Table 5.2: Pixel-wise objective evaluation of fish bone detection.

Testing image	Method	Precision	Recall	F-measure	IoU
No. 1	CNN-AFB	0.932	0.364	0.523	0.421
	CNN-VFB	0.739	0.762	0.751	0.664
No. 2	CNN-AFB	0.983	0.489	0.653	0.539
	CNN-VFB	0.713	0.817	0.761	0.655
No. 3	CNN-AFB	0.956	0.362	0.526	0.428
	CNN-VFB	0.641	0.744	0.689	0.579
No. 4	CNN-AFB	0.895	0.077	0.141	0.087
	CNN-VFB	0.700	0.733	0.716	0.623
No. 5	CNN-AFB	0.937	0.228	0.366	0.324
	CNN-VFB	0.821	0.648	0.724	0.680
No. 6	CNN-AFB	0.922	0.331	0.448	0.424
	CNN-VFB	0.739	0.781	0.759	0.669
No. 7	CNN-AFB	0.870	0.418	0.564	0.447
	CNN-VFB	0.690	0.808	0.744	0.637
No. 8	CNN-AFB	0.944	0.357	0.518	0.429
	CNN-VFB	0.831	0.799	0.814	0.751
No. 9	CNN-AFB	0.963	0.438	0.602	0.512
	CNN-VFB	0.790	0.737	0.763	0.694
No. 10	CNN-AFB	0.952	0.389	0.552	0.471
	CNN-VFB	0.719	0.774	0.745	0.652
Average	CNN-AFB	**0.935**	0.345	0.493	0.408
	CNN-VFB	0.738	**0.760**	**0.747**	**0.660**

IoU denotes intersection over union.

$$\text{AUC-PR} = \sum_{n} \frac{P_n + P_{n-1}}{2}(R_n - R_{n-1}), \tag{5.14}$$

where P_n and R_n are the precision and recall at the nth threshold. The interval of the threshold for calculating AUC-PR was 0.00025. As shown in Figure 5.8, the AUC-PR of CNN-VFB was 0.700, which is 0.163 point higher than that of CNN-AFB. Thus, proposed synthetic image

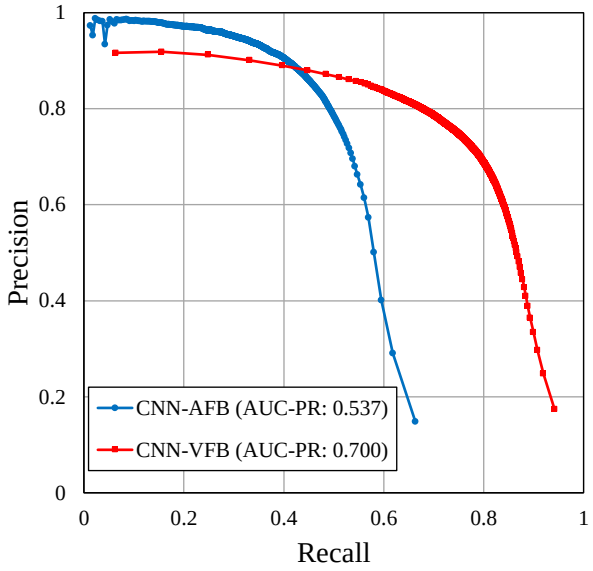


Figure 5.8: Precision-recall curves (PR curves) and the area of the under the precision-recall curve (AUC-PR). The AUC-PRs show that the detection performance of CNN-VFB is better than that of CNN-AFB.

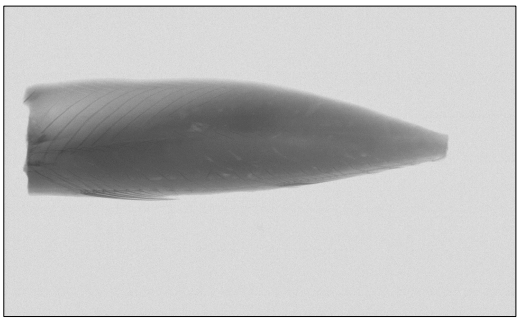
generation is very useful in cases wherein a large amount of training data cannot be prepared.

5.3.3 Subjective Evaluation of Detection Results

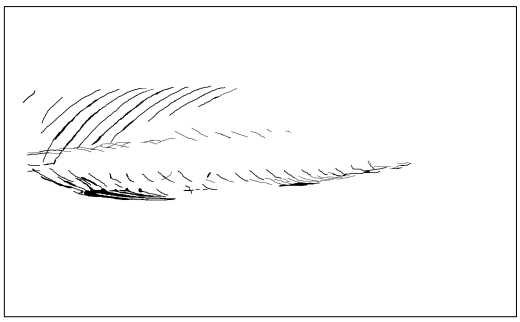
Figures 5.10 and 5.11 show detection results. CNN-VFB detected more fish bone patterns than CNN-AFB. Even CNN-VFB, however, cannot detect small bones. This is the reason why recall does not reach 0.9. Improvement of recall will be the next goal.

5.4 Conclusion

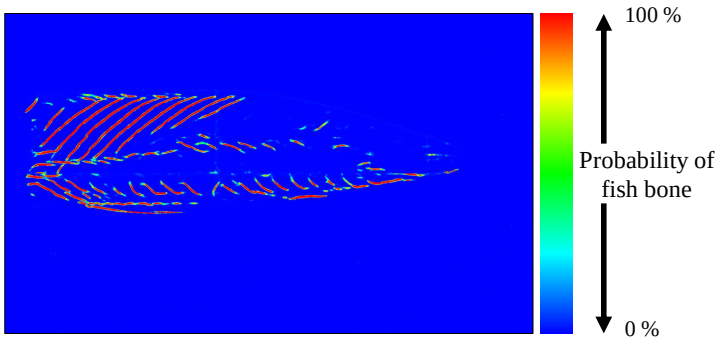
This chapter proposes a new fish bone detection technique using a CNN and synthetic image generation. The proposed method draws virtual fish bones on X-ray images. A total of 120 training images and their pixel-wise teaching signals were generated automatically from 12 X-ray images. Experimental results have shown that the average F-measure of the CNN trained with the 120 synthetic images was higher than that of the CNN trained with 10 images that included actual fish bones. This result suggests that it is possible to develop a high-accuracy fish



(a) Testing image



(b) ground truth



(c) heat map of fish bone probability

Figure 5.9: Testing image (No.5), its ground truth, and heat map of fish bone probability with CNN-VFB.

bone detector without manual annotation. This chapter ensured that proposed synthetic image generation is useful for reducing the cost of collecting and annotating a dataset. For applying proposed method to other fish types, however, it may be needed to adjust some parameters such as thickness and darkness of the virtual fish bones. Moreover, other bone patterns may improve the performance. Applying the proposed method to different types of fish and increasing the variations of virtual fish bones will be the scopes of the future works.

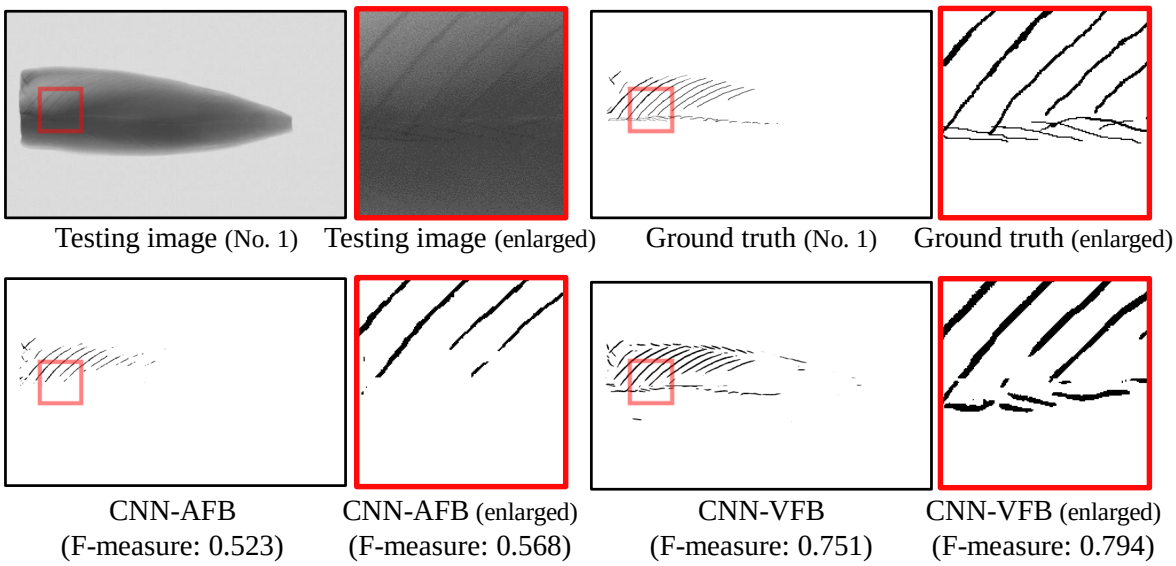


Figure 5.10: Detection results (Testing image No. 1).

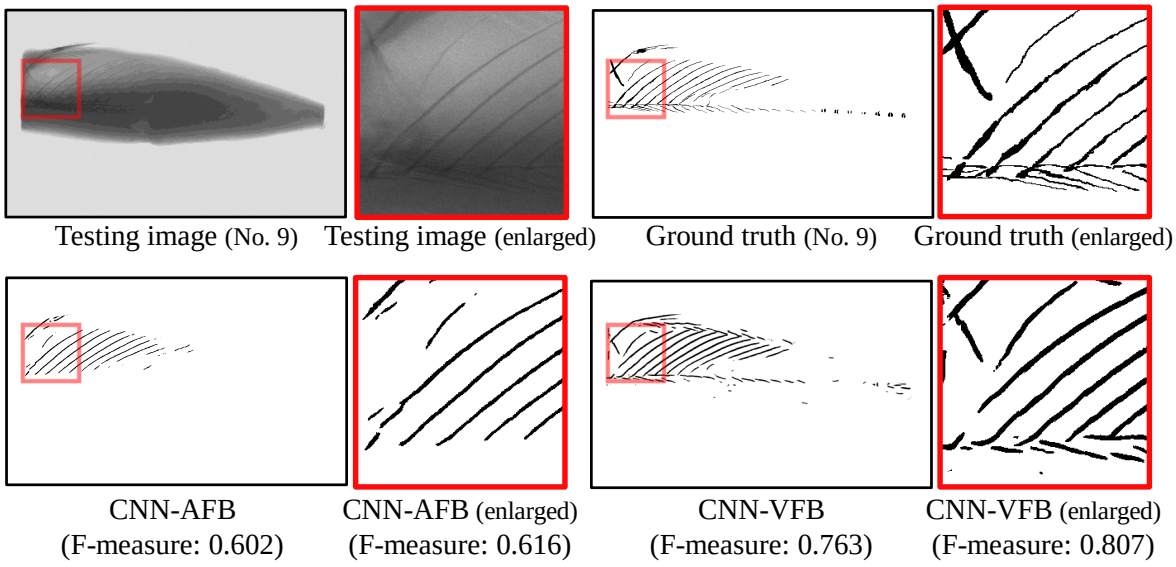


Figure 5.11: Detection results (Testing image No. 9).

6 Combination of Convolutional Neural Network Architecture and Its Learning Method for Rotation-Invariant Handwritten Digit Recognition

6.1 Introduction

Handwritten digit recognition is a classical and important task in computer vision. Recently, CNNs have achieved state-of-the-art results in digit recognition [62]. However, standard CNNs are not invariant to a large rotation transformation of input images. A simple solution for getting the rotation invariance is rotation data augmentation (RDA). RDA rotates original images and adds them to the original dataset. A common practice for data augmentation is to assign the same label to all augmented images of the same source. However, rotated digits seem to have different meanings in some cases. For example, the 180° rotated "6" looks like the positive "9". As augmented images compete with other augmented images, RDA may degrade the performance of digit recognition. Therefore, this chapter proposes a MTL for rotation invariance [63] and details the applicability for popular CNN architectures.

6.2 Related Works

Some studies have proposed rotation-invariant digit recognition using CNNs. Spatial transformer networks (STNs) [64] estimate the parameters of transformation from input feature maps and then spatially transform them. However, additional computations involved in the STN module makes the training of the CNN complex. Rotational invariance using multiple instances of CNN (RIMCNN) [65] trains a single CNN with single-orientation training images and then estimates testing images with multiple orientations. However, RIMCNN needs to infer a testing image multiple times. Reference [66] discussed the rotation invariance of natural image

classification. However, rotated natural images are not in conflict with each other unlike digit images. In contrast, this proposed method presents a single CNN and its learning method for rotation-invariant digit recognition.

6.3 Learning Methods of CNNs

6.3.1 Common Labeling of Data Augmentation

Figure 6.1-(a) shows common labeling of RDA. Let $\boldsymbol{x}$ be an input image including images augmented by RDA, $d \in \{0, 1, \cdots, 9\}$ be its digit label, and $r \in \{0, 1, \cdots, R-1\}$ be its rotation label, where R is the number of rotation orientations. Let L_{SCE} be the SoftMax cross entropy loss function and $F_{\mathrm{s}}(\boldsymbol{x}; \Theta)$ be an output of a NN after SoftMax function with the parameter Θ. The loss of a common label is calculated as follows:

$$L_{\mathrm{RDA}}(\Theta) = L_{\mathrm{SCE}}(F_{\mathrm{s}}(\boldsymbol{x}; \Theta), d). \tag{6.1}$$

All augmented images of the same source are assigned the same label. As there are no labels for rotation in Equation. 6.1, understanding rotation transformation is difficult for CNNs.

6.3.2 Joint-Label Classifier

Figure 6.1-(b) shows the overview of joint-label classifier (JLC) [66]. The loss of JLC is calculated as follows:

$$L_{\mathrm{JLC}}(\Theta) = L_{\mathrm{SCE}}(F_{\mathrm{s}}(\boldsymbol{x}; \Theta), (d, r)). \tag{6.2}$$

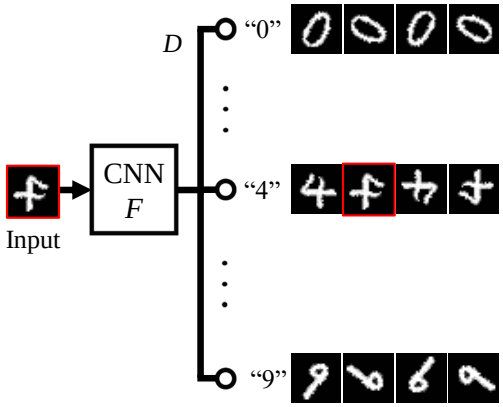
The number of output units is $10R$. JCL can learn the joint probability distribution on all combinations of digit and rotation labels.

6.3.3 Multi-Task Learning

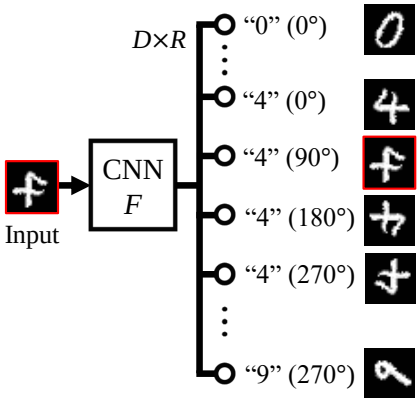
Figure 6.1-(c) shows the overview of MTL [40]. The loss of MTL is calculated as follows:

$$L_{\mathrm{MTL}}(\Theta) = L_{\mathrm{SCE}}(F_{\mathrm{s.D}}(\boldsymbol{x}; \Theta), d) + w_{\mathrm{rot}} L_{\mathrm{SCE}}(F_{\mathrm{s.R}}(\boldsymbol{x}; \Theta), r). \tag{6.3}$$

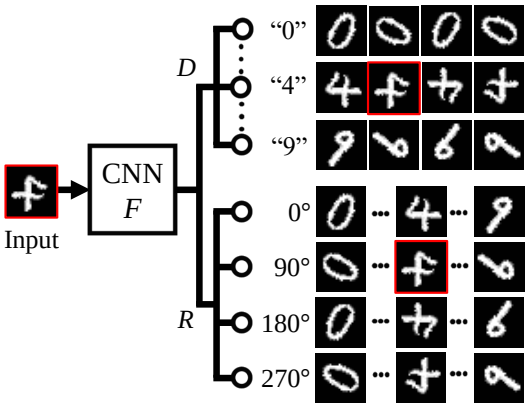
The numbers of output units for $F_{\mathrm{s.D}}$ and $F_{\mathrm{s.R}}$ are 10 and R, respectively. MTL improves the main-task performance by using the optional information contained in sub-tasks. In contrast to JCL, MTL can control the balance of the main as well as sub-tasks using weight factor w_{rot}.



(a) Rotation data augmentation (RDA).



(b) RDA + Joint-label classifier (JLC).



(c) RDA + Multi-task learning (MTL).

Figure 6.1: Learning methods for rotation invariant-digit recognition.

Table 6.1: Dependence of weight factor w_{rot} on MTL.

CNN architecture	Top-1 Accuracy [%]			
	$w_{\text{rot}}=1$	$w_{\text{rot}}=0.5$	$w_{\text{rot}}=0.25$	$w_{\text{rot}}=0.1$
LeNet-5 [67]	98.93	98.98	**99.02**	98.93
ResNet-18 [21]	99.45	**99.50**	**99.50**	99.42
ResNet-50 [21]	99.56	99.56	99.59	**99.61**
WideResNet 16-8 [68]	**99.52**	99.46	99.44	99.31
WideResNet 28-10 [68]	**99.59**	99.58	99.56	99.55

Bold red font indicates the best score in each CNN architecture.

6.4 Implementation Details

This chapter uses a handwritten digit dataset called MNIST with 60,000 training images and 10,000 testing images [62, 67]. These images are augmented 4 times with rotation angles of 0°, 90°, 180°, and 270°. Three types of learning methods stated in Section 6.3 are applied to five popular CNN architectures [21,67,68]. All CNNs are implemented using a Pytorch package [69]. According to Reference [68], the initial learning rate is set to 0.1 and dropped by 0.2 at 60, 120, and 160 epochs, and the total epoch is 200.

6.5 Evaluation Experiments

Table 6.1 shows the dependence of weight factor w_{rot} on MTL. The weight factor for the best top-1 accuracy is dependent on the CNN architecture. Thus, MTL needs to find a weight factor for the best performance.

Table 6.2 shows combinations of CNN architectures and their learning methods. All accuracies are not for the rotation labels, but for the digit labels. Both JLC and MTL improved the top-1 accuracies of RDA. This result implies that learning the rotation information is important for rotation-invariant digit recognition.

Figure 6.2 shows the loss curves on ResNet-50. In Table 6.2 and Figure 6.2, MTL performed with the highest top-1 accuracy and the lowest test loss on ResNet-50. Table 6.3 shows the result for each digit on ResNet-50. MTL is especially effective for rotation-dependent digits such as "6". This chapter concluded that MTL on ResNet-50 is the best solution for rotation-invariant digit recognition.

Table 6.2: Combinations of CNN architecture and the learning method.

CNN architecture	Top-1 Accuracy [%] / Test Loss		
	RDA	RDA+JLC	RDA+MTL
LeNet-5 [67]	98.91 / 0.0374	99.00 / 0.0406	**99.02** / **0.0356**
ResNet-18 [21]	99.37 / 0.0216	99.47 / 0.0200	**99.50** / **0.0179**
ResNet-50 [21]	99.44 / 0.0200	99.51 / 0.0175	**99.61** / **0.0155**
WideResNet 16-8 [68]	99.13 / 0.0316	99.50 / **0.0200**	**99.52** / 0.0221
WideResNet 28-10 [68]	99.47 / 0.0181	99.53 / 0.0214	**99.59** / **0.0167**

Bold red and underlined blue fonts indicate the best and second best scores, respectively.

Table 6.3: Effectiveness for each digit on ResNet-50.

Digit	Top-1 Accuracy [%] / Test Loss		
	RDA	RDA+JLC	RDA+MTL
“0”	99.74 / 0.0089	99.77 / 0.0073	99.87 / 0.0044
“1”	99.80 / 0.0066	99.82 / 0.0059	99.82 / 0.0076
“2”	99.66 / 0.0183	99.66 / 0.0131	99.59 / 0.0160
“3”	99.55 / 0.0160	99.75 / 0.0083	99.85 / 0.0055
“4”	99.47 / 0.0201	99.59 / 0.0166	99.72 / 0.0126
“5”	98.87 / 0.0447	99.13 / 0.0302	99.19 / 0.0253
“6”	98.98 / 0.0414	99.03 / 0.0333	99.45 / 0.0241
“7”	99.56 / 0.0304	99.44 / 0.0245	99.64 / 0.0182
“8”	99.67 / 0.0115	99.64 / 0.0120	99.71 / 0.0122
“9”	99.01 / 0.0366	99.15 / 0.0334	99.18 / 0.0310

Learning rotation information is effective for rotation-dependent digits such as “6”.

6.6 Conclusion

This chapter presents combinations of CNNs and their learning methods for rotation-invariant handwritten digit recognition. Experimental results have shown that MTL on ResNet-50 is the best solution for that. Implementing free rotation from 0° to 360° will be the scope of the future works.

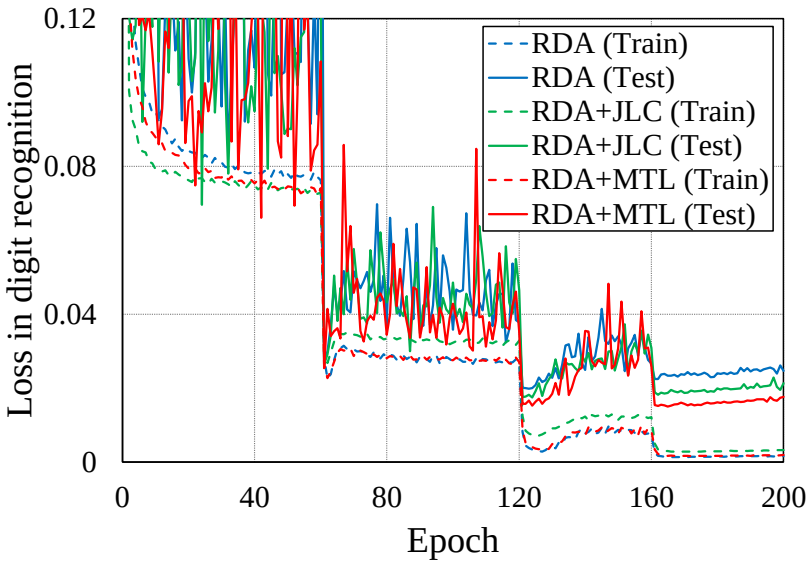


Figure 6.2: Loss curves on ResNet-50.

7 Conclusion

This dissertation proposed the following important studies for the efficient learning of neural networks in image processing:

1. Image Super-Resolution with Multi-Path Convolutional Neural Network (Chapter 2)
2. Luminance Isotropy Improvement for Convolutional Neural Network-Based Image Super-Resolution (Chatpter 3)
3. Multi-Category Image Super-Resolution with Convolutional Neural Network and Multi-Task Learning (Chapter 4)
4. Automated Fish Bone Detection in X-ray Images with Convolutional Neural Network and Synthetic Image Generation (Chapter 5)
5. Combination of Convolutional Neural Network Architecture and Its Learning Method for Rotation-Invariant Handwritten Digit Recognition (Chapter 6)

Chapter 2 proposed super-resolutions using a CNN with multiple propagation paths for high-quality and high-speed resolution. In the proposed method, a CNN has multiple paths from low to deep layers, and generates low- and high-frequency signals independently. In the evaluation results, the processing time of the proposed method was reduced to approximately a quarter of that of conventional MCH with the same PSNR. These results show that the proposed architecture with multiple propagation paths is useful. The proposed method helps introduce CNN-based super-resolutions to televisions and smartphones without using accelerators such as GPUs.

Chapter 3 proposed the "luminance inversion averaging" method to improve the luminance isotropy of CNN-based image super-resolutions. The results show that the average PSNR for super-resolution using luminance inversion averaging is approximately 0.15–0.20 dB higher than that of conventional super-resolution. This chapter verified that the proposed method can

improve the performances of CNN-based super-resolutions even if the luminance distributions of training and inferred images differ from each other.

Chapter 4 proposed an image super-resolution for multiple image categories with a single CNN and MTL. The proposed CNN does not require an external category of information; however, it has an internal category-classifying ability. In the experiments, the average PSNR of the proposed method was approximately 0.22 dB higher than that of the conventional method with the same training dataset and processing time. The proposed method is useful when the input image category varies.

Chapter 5 proposed a new fish bone detection technique using a CNN and synthetic image generation. The proposed method draws virtual fish bones on X-ray images. The results show that the average F-measure of the CNN trained with 120 synthetic images was higher than that of the CNN trained with 10 images that included actual fish bones. This suggests that developing a high-accuracy fish bone detector is possible without manual annotation. This chapter verifies that the proposed synthetic image generation is useful for reducing the cost of collecting and annotating a dataset.

Chapter 6 presented combinations of CNNs and their learning methods for rotation-invariant handwritten digit recognition. The results show that MTL on ResNet-50 is the best solution.

Through the above five chapters, this dissertation emphasized the importance of the following three factors:

(i) image dataset

(ii) network architecture

(iii) learning algorithm

Regarding (i) image dataset, this study has shown that virtual samples generated using computer graphics are efficient for use in neural networks. Since automated image generation can reduce the cost of dataset collection and annotation, it is useful for generating rare scenes and covering various conditions. For example, medical images include rare scenes that are uncollectable. In addition, autonomous driving needs to cover a wide range of use cases. Computer graphics can simulate rare scenes and various conditions with less cost and thus improve detection performance. This study has also shown that virtual samples cannot necessarily simulate the details of real ones and as such, although it is difficult to model the details specifically, the virtual targets are designed using domain knowledge. Therefore, a combination of virtual and

real samples is a good solution. To produce a good quality image dataset at a reduced cost is a challenging theme in the field of neural network-based image processing.

Regarding (ii) network architecture, this study has shown that a design based on the frequency components and feature extraction can reduce processing time and improve robustness. The network architecture is especially important for implementing image processing on edge devices and limited resources should be used efficiently. Since the optimal architectures depend on the hardware specification, they may need to be designed individually. Therefore, an automated system that can identify the optimal network architecture is one of the desirable solutions. An efficient network architecture design will be a worthwhile theme for engineers of neural network-based image processing.

Regarding (iii) learning algorithm, this study has shown that the algorithm using relevant tasks or additional information improves robustness, which is required to apply neural network-based image processing to various use cases and tolerate unexpected input. This study has also shown that performance may degrade for the specific tasks and that the annotation cost of a training dataset could increase. Therefore, the definition of use cases and coverage is important to control the trade-off issue. In addition, a learning algorithm that uses existing information with a reduced annotation cost is useful. An efficient learning algorithm with high robustness and reduced collection cost will be a demanding point for user-friendly systems with neural network-based image processing.

In conclusion, for an efficient learning method, (i) image dataset, (ii) network architecture, and (iii) learning algorithm are essential factors for improving the performance of neural network based-image processing.

This dissertation presented valuable knowledge and ideas for the efficient learning of neural networks for image processing. Image processing techniques are used throughout life and society. I hope my studies aid the development of image processing and lead to happier lives.

References

[1] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, 2012.

[2] C. Dong, C.C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in Computer Vision – ECCV 2014, pp.184–199, 2014.

[3] C. Dong, C.C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.38, no.2, pp.295–307, 2016.

[4] W.T. Freeman, "Learning low-level vision," International Journal of Computer Vision, vol.40, no.1, pp.25–47, 2000.

[5] R. Timofte, V.D. Smet, and L.V. Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in Computer Vision – ACCV 2014, pp.111–126, 2015.

[6] J. Salvador and E. Perez-Pellitero, "Naive bayes super-resolution forest," 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[7] E. Perez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn, "PSyCo: Manifold span reduction for super resolution," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[8] Y. Tian, F. Zhou, W. Yang, X. Shang, and Q. Liao, "Anchored neighborhood regression based single image super-resolution from self-examples," 2016 IEEE International Conference on Image Processing (ICIP), 2016.

[9] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[10] J. Kim, J.K. Lee, and K.M. Lee, "Accurate image super-resolution using very deep convolutional networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[11] C. Dong, C.C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in Computer Vision – ECCV 2016, pp.391–407, 2016.

[12] W. Shi, J. Caballero, F. Huszar, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[13] Y. Kato, S. Ohtani, N. Kuroki, T. Hirose, and M. Numa, "Image super-resolution with multi-channel convolutional neural networks," 14th IEEE International New Circuits and Systems Conference (NEWCAS), 2016.

[14] S. Ohtani, Y. Kato, N. Kuroki, T. Hirose, and M. Numa, "Multi-channel convolutional neural networks for image super-resolution," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E100.A, no.2, pp.572–580, 2017.

[15] Y. Kato, S. Ohtani, N. Kuroki, T. Hirose, and M. Numa, "Super-resolution with horizontal and vertical convolutional neural networks," IEEJ Transactions on Electronics, Information and Systems, vol.138, no.7, pp.957–963, 2018. (in japanese).

[16] W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," IEEE Transactions on Multimedia, vol.21, no.12, pp.3106–3121, 2019.

[17] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[18] V. Nair and G.E. Hinton, "Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th International Conference on International Conference on Machine Learning, pp.807–814, 2010.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, pp.448–456, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," Proceedings of the 22nd ACM International Conference on Multimedia, pp.675–678, 2014.

[23] H.T. Tran and T. Ho-Phuoc, "Deep laplacian pyramid network for text images super-resolution," 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF), 2019.

[24] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[25] Y. Shi, K. Wang, C. Chen, L. Xu, and L. Lin, "Structure-preserving image super-resolution via contextualized multitask learning," IEEE Transactions on Multimedia, vol.19, no.12, pp.2804–2815, 2017.

[26] K. Urazoe, N. Kuroki, Y. Kato, S. Ohtani, T. Hirose, and M. Numa, "Super-resolution with multi-path convolutional neural networks," IEEJ Transactions on Electronics, Information and Systems, vol.140, no.6, pp.638–650, 2020. (in japanese).

[27] K. Urazoe, N. Kuroki, Y. Kato, S. Ohtani, T. Hirose, and M. Numa, "Improvement of luminance isotropy for convolutional neural networks-based image super-resolution," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E103.A, no.7, pp.955–958, 2020.

[28] J. Yang, J. Wright, T.S. Huang, and Y. Ma, "Image super-resolution via sparse representation," IEEE Transactions on Image Processing, vol.19, no.11, pp.2861–2873, 2010.

[29] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," Proceedings Eighth IEEE International Conference on Computer Vision (ICCV), 2001.

[30] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," Procedings of the British Machine Vision Conference, 2012.

[31] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," Proceedings of the 7th International Conference on Curves and Surfaces, pp.711–730, 2012.

[32] W.S. Lai, J.B. Huang, N. Ahuja, and M.H. Yang, "Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[33] Z. Wang, J. Chen, and S.C.H. Hoi, "Deep learning for image super-resolution: A survey," arXiv: abs/1902.06068, 2019.

[34] J.B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[35] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," Multimedia Tools Applications., vol.76, no.20, pp.21811–21838, 2017.

[36] T. Ogawa, A. Otsubo, R. Narita, Y. Matsui, T. Yamasaki, and K. Aizawa, "Object detection for comics using manga109 annotations," arXiv: abs/1803.08670, 2018.

[37] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," Proceedings of the 23rd ACM International Conference on Multimedia, pp.689–692, 2015.

[38] V.K. Ha, J.C. Ren, X.Y. Xu, S. Zhao, G. Xie, V. Masero, and A. Hussain, "Deep learning based single image super-resolution: A survey," International Journal of Automation and Computing, vol.16, no.4, pp.413–426, 2019.

[39] M. Hradiš, J. Kotera, P. Zemčík, and F. Šroubek, "Convolutional neural networks for direct text deblurring," Proceedings of the British Machine Vision Conference, 2015.

[40] R. Caruana, "Multitask learning," Machine Learning, vol.28, no.1, pp.41–75, 1997.

[41] Y. Zhang and Q. Yang, "A survey on multi-task learning," arXiv: abs/1707.08114, 2017.

[42] A.H. Abdulnabi, G. Wang, J. Lu, and K. Jia, "Multi-task CNN model for attribute prediction," IEEE Transactions on Multimedia, vol.17, no.11, pp.1949–1959, 2015.

[43] M.S. Rad, B. Bozorgtabar, C. Musat, U.V. Marti, M. Basler, H.K. Ekenel, and J.P. Thiran, "Benefiting from multitask learning to improve single image super-resolution," Neurocomputing, 2019.

[44] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-stuff: Thing and stuff classes in context," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[45] M. Lin, Q. Chen, and S. Yan, "Network in network," 2nd International Conference on Learning Representations, ICLR, 2014.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol.15, pp.1929–1958, 2014.

[47] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in Advances in Neural Information Processing Systems 27, pp.3320–3328, 2014.

[48] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations (ICLR), 2015.

[49] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[50] Q. Chen, C. Zhang, J. Zhao, and Q. Ouyang, "Recent advances in emerging imaging techniques for non-destructive detection of food quality and safety," TrAC Trends in Analytical Chemistry, vol.52, pp.261–274, 2013.

[51] D. Mery, I. Lillo, H. Loebel, V. Riffo, A. Soto, A. Cipriano, and J.M. Aguilera, "Automated fish bone detection using x-ray imaging," J. of Food Engineering, vol.105, no.3, pp.485–492, 2011.

[52] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.

[53] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," Medical Image Computing and Computer-Assisted Intervention, pp.234–241, 2015.

[54] F. Lateef and Y. Ruichek, "Survey on semantic segmentation using deep learning techniques," Neurocomputing, vol.338, pp.321–348, 2019.

[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv, abs/1409.1556, 2014.

[56] M. Everingham, S.M.A. Eslami, L.V. Gool, C.K.I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," International Journal of Computer Vision, vol.111, no.1, pp.98–136, 2014.

[57] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, "Microsoft COCO: Common objects in context," in Computer Vision – ECCV 2014, pp.740–755, 2014.

[58] Ishida Co., Ltd, "Xray-detector IX-GN." https://www.ishida.co.jp/ww/jp/products/documents/upload/ix-gn.pdf. (accessed Oct. 15, 2020).

[59] C. Shorten and T.M. Khoshgoftaar, "A survey on image data augmentation for deep learning," J. of Big Data, vol.6, no.1, 2019.

[60] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp.249–256, 2010.

[61] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," Proceedings of the 23rd international conference on Machine learning (ICML), 2006.

[62] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with MNIST and EMNIST," Applied Sciences, vol.9, no.15, p.3169, 2019.

[63] K. Urazoe, N. Kuroki, T. Hirose, and M. Numa, "Rotation invariant-digits recognition with single convolutional neural networks," Proceedings of 2020 RISP International Workshop

on Nonlinear Circuits, Communications and Signal Processing (NCSP 2020), pp.618–621, 2020.

[64] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," Advances in Neural Information Processing Systems, 2015.

[65] A. Jain, G.R.K.S. Subrahmanyam, and D. Mishra, "Rotation invariant digit recognition using convolutional neural network," in Proceedings of 2nd International Conference on Computer Vision & Image Processing, pp.91–102, 2018.

[66] H. Lee, S.J. Hwang, and J. Shin, "Rethinking data augmentation: Self-supervision and self-distillation," arXiv: abs/1910.05872, 2019.

[67] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol.86, no.11, pp.2278–2324, 1998.

[68] S. Zagoruyko and N. Komodakis, "Wide residual networks," Procedings of the British Machine Vision Conference 2016, British Machine Vision Association, 2016.

[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," Advances in Neural Information Processing Systems, 2019.

Acknowledgment

I would like to express my sincere gratitude to Associate Prof. Nobutaka Kuroki, Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University, for his kind guidance and encouragement in conducting out this study. Thank you from the bottom of my heart.

Prof. Masahiro Numa, Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University, provides me with valuable advice and guidance in this study, and I deeply appreciate his input.

Prof. Tetsuya Hirose, Division of Electrical, Electronic and Infocommunications Engineering, Graduate School of Engineering, Osaka University, gives me warm encouragement and valuable opinions, and I thank for his commitments.

Profs. Tsutomu Terada and Masahide Nakamura, Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University, review this dissertation, and I express a special thanks for their contributions.

Ms. Kaori Matsumoto, the technical expert, Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University, provides a good laboratory environment, and I extend my sincere thanks for her supports.

I also thank the members of Integrated Circuits and Information Laboratory, Department of Electrical and Electronic Engineering, Graduate School of Engineering, Kobe University, and offer a special thanks to Dr. Yu Kato and Mr. Shinya Otani for their insightful discussions and studies.

The study in Chapter 5 was conducted in the collaboration with Ishida Co., Ltd. Mr. Mizuki Iwabuchi, Mr. Akihiro Maenaka, Mr. Hironori Tsutsumi, and Mr. Kosuke Fuchuya provide their important discussions, contributions, and a considerable amount of the dataset, and I thank for their works.

Finally, I would like to express my deep gratitude towards my family for their warm support.

Publications

Journals

1. Kazuya Urazoe, Nobutaka Kuroki, Yu Kato, Shinya Ohtani, Tetsuya Hirose, Masahiro Numa, "Super-resolution with multi-path convolutional neural networks," IEEJ Transactions on Electronics, Information and Systems, vol. 140, no. 6, pp. 638–650, 2020. (in japanese)
   浦添和哉, 黒木修隆, 加藤裕, 大谷真也, 廣瀬哲也, 沼昌宏, "複数の伝搬経路を持つ畳み込みニューラルワークによる超解像," 電気学会論文誌C（電子・情報・システム部門誌）, vol. 140, no. 6, pp. 638–650, 2020.

2. Kazuya Urazoe, Nobutaka Kuroki, Yu Kato, Shinya Ohtani, Tetsuya Hirose, Masahiro Numa, "Improvement of luminance isotropy for convolutional neural networks-based image super-resolution," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E103-A, no. 7, pp. 955–958, 2020.

3. Kazuya Urazoe, Nobutaka Kuroki, Tetsuya Hirose, Masahiro Numa, "Combination of convolutional neural network architecture and its learning method for rotation-invariant handwritten digit recognition," IEEJ Transactions on Electrical and Electronic Engineering, vol. 16, no. 1, pp. 161–163, 2021.

4. Kazuya Urazoe, Nobutaka Kuroki, Yu Kato, Shinya Ohtani, Tetsuya Hirose, Masahiro Numa, "Multi-category image super-resolution with convolutional neural network and multi-task learning," IEICE Transactions on Information and Systems, vol. E104-D, no. 1, pp. 183–193, 2021.

5. Kazuya Urazoe, Nobutaka Kuroki, Akihiro Maenaka, Hironori Tsutsumi, Mizuki Iwabuchi, Tetsuya Hirose, Masahiro Numa, "Automated fish bone detection in X-ray images with convolutional neural network and synthetic image generation," IEEJ Transactions on Elec-

trical and Electronic Engineering, vol. 16, no. 11, pp. 1510–1517, 2021.

International Conferences

1. Kazuya Urazoe, Nobutaka Kuroki, Tetsuya Hirose, Masahiro Numa, "Rotation invariant-digits recognition with single convolutional neural networks," Proceedings of 2020 RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing (NCSP 2020), 2PM2-1-4, pp. 618–621, Hawaii, USA, Feb. 28–Mar. 2, 2020.

Domestic Conferences

1. Kazuya Urazoe, Nobutaka Kuroki, Tetsuya Hirose, Masahiro Numa, "Automatic image generation by DCGAN with semantic segmentation", Forum on Information Technology (FIT2018), H-023, vol. 3, pp. 135–136, Fukuoka, Sept. 19–21, 2018. (in japanese)
浦添和哉, 黒木修隆, 廣瀬哲也, 沼昌宏, "セグメンテーションを組み合わせた DCGAN による画像の自動生成," 第 17 回情報科学技術フォーラム (FIT2018), H-023, 第 3 分冊, pp. 135–136, 福岡, 2018 年 9 月 19–21 日.

2. Kazuya Urazoe, Nobutaka Kuroki, Yu Kato, Shinya Ohtani, Tetsuya Hirose, Masahiro Numa, "A tiny convolutional neural network for image super-resolution," IEICE Technical Report, vol. 121, no. 217, IMQ2021-7, pp. 2-7, Oct. 22, 2021.
浦添和哉, 黒木修隆, 加藤裕, 大谷真也, 廣瀬哲也, 沼昌宏, "A tiny convolutional neural network for image super-resolution," 電子情報通信学会 技術研究報告, vol. 121, no. 217, IMQ2021-7, pp. 2-7, 2021 年 10 月 22 日.

Awards

1. 浦添和哉, "セグメンテーションを組み合わせた DCGAN による画像の自動生成," 第 17 回情報科学技術フォーラム (FIT2018), FIT 奨励賞, 2018 年 9 月 21 日.

2. 浦添和哉, "畳み込みニューラルネットワークと生成型学習を用いた食品内の微細異物検出," 神戸大学竹水会 竹水会優秀論文賞, 2020 年 3 月 25 日.

Doctor Thesis, Kobe University
"Efficient Learning of Neural Networks for Image Processing," 104 pages.
Submitted on January, 17, 2023.

The date of publication is printed in cover of repository version published in Kobe University Repository Kernel.