Ph.D. DISSERTATION

# Image restoration using neural architecture search method

## 심층 신경망 검색 기법을 사용한 이미지 복원

BY

AHN JOON YOUNG

AUGUST 2021

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

**Image restoration using neural architecture search method**

**심층 신경망 검색 기법을 사용한 이미지 복원**

지도교수 조 남 익

이 논문을 공학박사 학위논문으로 제출함
2021 년 7 월

서울대학교 대학원
전기·정보공학부
안 준 영

안준영의 공학박사 학위논문을 인준함
2021 년 6 월

위 원 장 _____

부위원장 _____

위 원 _____

위 원 _____

위 원 _____

# Abstract

Image restoration is an important technology which can be used as a pre-processing step to increase the performances of various vision tasks. Image super-resolution is one of the important task in image restoration which restores a high-resolution (HR) image from low-resolution (LR) observation. The recent progress of deep convolutional neural networks has enabled great success in single image super-resolution (SISR). its performance is also being increased by deepening the networks and developing more sophisticated network structures. However, finding an optimal structure for the given problem is a difficult task, even for human experts. For this reason, neural architecture search (NAS) methods have been introduced, which automate the procedure of constructing the structures. In this dissertation, I propose a new single image super-resolution framework by using neural architecture search (NAS) method. As the performance improves, the network becomes more complex and deeper, so I apply NAS algorithm to find the optimal network while reducing the effort in network design. In detail, the proposed scheme is summarized to three topics: image super-resolution using efficient neural architecture search, multi-branch neural architecture search for lightweight image super-resolution, and neural architecture search for image super-resolution using meta-transfer learning.

At first, I expand the NAS to the super-resolution domain and find a lightweight densely connected network named DeCoNASNet. I use a hierarchical search strategy to find the best connection with local and global features. In this process, I define a complexity-based-penalty and add it to the reward term of REINFORCE algorithm. Experiments show that my DeCoNASNet outperforms the state-of-the-art lightweight super-resolution networks designed by handcraft methods and existing NAS-based design.

I propose a new search space design with multi-branch structure to enlarge the

search space for capturing multi-scale features, resulting in better reconstruction on grainy areas. I also adopt parameter sharing scheme in multi-branch network to share their information and reduce the whole network parameter. Experiments show that the proposed method finds an optimal SISR network about twenty times faster than the existing methods, while showing comparable performance in terms of PSNR vs. parameters. Comparison of visual quality validates that the proposed SISR network reconstructs texture areas better than the previous methods because of the enlarged search space to find multi-scale features.

Lastly, I apply meta-transfer learning to the NAS procedure for image super-resolution. I train the controller and child network with the meta-learning scheme, which enables the controllers to find promising network for several scale simultaneously. Furthermore, meta-trained child network is reused as the pre-trained parameters for final evaluation phase to improve the final image super-resolution results even better and search-evaluation gap problem is efficiently reduced.

**keywords**: Neural architecture search, image restoration, image super-resolution, deep learning

**student number**: 2014-22565

# Contents

# List of Tables

# List of Figures

# Chapter 1

# INTRODUCTION

Nowadays, numerous images and videos are taken from smartphones and shared to the internet. Due to the lack of mobile data, images and videos are compressed or resized when they are uploaded to SNS. Once the image is resized to small size, the information is disappeared and they can't be up-sampled simply. The image super-resolution algorithms are used in this situation, to reconstruct the high-resolution image from low-resolution input. It is also an important technology that can be widely used as a pre-processing step of various tasks such as medical image analysis [8], satellite image recognition [9], security image processing [10], etc. with the development of deep learning area, the SISR algorithms also change their paradigm from conventional interpolation-based [11] or reconstructuion-based method [12] to learning-based methods which train the neural networks with many LR-HR image pairs to create high-resolution images.

Many deep neural networks for SISR have been proposed [1–5, 13–18]. The convolution layer and its variants such as dilated convolution or depth-separable convolution is used as a basic operation of the deep learning-based algorithms. Researchers have proposed specific structures to achieve better performance.

However, in designing a deep network, I should select a considerable number of network configurations such as connection, operation type, the number of feature chan-

nels, depth, etc. Researchers have designed their structures through a large number of trials to achieve a competent performance. It is a tedious task and difficult to find an optimal system for a given task. The NAS algorithms have been proposed to alleviate this burden, especially in the case of image classification researches [19–26].

Some researchers expanded the neural architecture search scheme to the other tasks such as object detection (Auto-deeplab) [27] and image SR (MoreMNAS, FALSR) [6, 7]. Specifically, they used a reinforced evolution algorithm and solved the image SR task as a multi-objective problem. However, these reinforced evolution algorithms need more than 1 GPU month to find optimal architectures. Furthermore, FALSR evaluated the network approximately because they did not use a complete training strategy.

To overcome these drawbacks, I propose a new NAS-based SR method by taking the idea of ENAS [21] as my baseline search framework. The ENAS consists of two components: a controller and child network. The controller is composed of LSTM blocks to generate a sequence used to construct child network. The REINFORCE [28] algorithm is used to train the controller to create optimal child network which results in high-quality SR results. As in the original ENAS for the classification problems, child networks share their parameters during the training and evaluation. In addition, I propose a complexity-based penalty to reduce the rewards of the networks that need a large number of parameters. I exclude redundant hierarchical information by predicting connections for local and global feature fusion through the controller. I Also create the densely connected search space on the baseline of residual dense network (RDN) [17]. The proposed search space consists of mix nodes for densely connected network blocks (DNB), local feature fusion, and global feature fusion. In addition to the convolution layer with $3times3$ filters in the baseline network, dilated convolution [29] and depth separable convolution [30] are included in the search space.

To further achieve better performance, I propose a new search space for image super-resolution. I find that unlike most of earlier learning-based SR methods which used single-branch network, some methods [15, 18, 31, 32] constructed multi-branch

networks to extract multi-scale features from low-resolution input. These methods tend to perform better with fewer parameters than single-branch networks. Based on this, in this dissertation, I propose an automated multi-branch search space design for the neural architecture search (NAS) scheme. To be specific, I propose a Multi-Branch Neural Architecture Search (MBNAS) algorithm, which tries to find optimal connections in multi-branch structures. The MBNAS search space consists of partially shared nodes (PSN) for multi-scale block, local feature fusion layer, and global feature fusion layer. The PSNs share their parameters with different branches to transmit information efficiently to them. For simplicity, I use only $3 \times 3$ convolution and $3 \times 3$ dilated convolution as the basic operation, and let the search algorithm find optimal connections.

Lastly, I adopt meta transfer-learning scheme to my NAS-SR method to conduct simultaneous search for image super-resolution of various scales. Existing NAS-SR methods can only perform network search for one scale at a time. Unlike previous methods, I train the controller and the child network with meta-transfer learning scheme. After meta-training, the parameters of the controller are able to adapt rapidly to the image super-resolution task of specific scale. Furthermore, the parameters of the child network can be used as the promising initial point for final evaluation and achieve better result.

## 1.1 contribution

The main contributions of this dissertation are summarized as follows:

1. New NAS-based SR:I propose a new NAS-based SR network design, which searches for networks with higher performance by combining hierarchical and local information efficiently.

2. Complexity-based penalty: I propose a complexity-based penalty and add it to the reward signal of the REINFORCE algorithm. This enables us to search for an efficient network that has high performance with a lightweight structure.

3. Feature Fusion Layer Search: I search for connection of feature fusion layer. instead of connecting all the global/local feature fusion layers. I design the connection to be predicted through the controller, which removes redundant hierarchical information and hence reduce network complexity.

4. Multi-branch structure for multi-scale feature extraction: I construct the network with a multi-branch structure, and each branch learns how to restore patterns of different scales.

5. Partially shared node (PSN): I partially share the parameters of branches to connect each other's information and construct a lightweight structure. The partially shared structure efficiently reduces the searched network's parameter without performance degradation.

6. Meta-transfer learning for simultaneous search of various scale image super-resolution: I apply meta-transfer learning to the controller and the child network of my framework, and find the promising structure for image SR network of various scale at the same time.

## 1.2 contents

The rest of this dissertation is organized as follows. In chapter 2, the proposed NAS-based image SR framework with single-branch structure, DeCoNAS is explained. I search for multi-branch image SR network named as MBNASNet in chapter 3. In chapter 4, I applied meta-learning scheme to the NAS-SR framework and achieve the simultaneous architecture search for various scale image super-resolution. Finally, this dissertation is concluded in chapter 5.

# Chapter 2

# Neural Architecture Search for Image Super-Resolution Using Densely Constructed Search Space: DeCoNAS

## 2.1 Introduction

Single image super-resolution (SISR) is a task that creates a clearer high-resolution image from a single low-resolution input. It is an important technology that can be used as a pre-processing step to increase performances of various tasks such as medical image analysis [8], satellite image recognition [9], security image processing [10], etc. The SISR is an ill-posed problem because multiple HR images can be mapped to a single LR image. Hence, learning-based methods trained with many LR-HR image pairs are generally more effective than the interpolation-based [11] or reconstruction-based methods [12].

Recently, many deep neural networks for SISR have been developed [1–5, 13–18], where Dong *et al.*'s SRCNN [1] is the first convolutional neural network (CNN) for the SISR. It consists of three convolution layers and yet outperformed conventional non-learning methods by a large margin. FSRCNN [13] and ESPCN [14] tried to reduce the computational cost of the structure. They used LR images directly as the input of their neural networks. Then, deconvolution and sub-pixel convolution layers were

used for upsampling their results. VDSR [2] dramatically increased the depth of the model by residual learning and gradient clipping strategy. Lim *et al.* [15] further improved performance by a residual block composed of extensive features (EDSR) and multi-scale structure (MDSR). In MemNet [4], MSRN [18], and DenseSR [16], they proposed specific blocks in their models, such as memory block, multi-scale residual block or dense block. SelNet [33] used selection unit instead of conventional Relu operation. Zhang *et al.* proposed RDN [17], which consists of residual dense block and dense feature fusion that extract abundant information from the input. RCAN [34] applied channel attention mechanism to improve representational ability of CNNs. It is believed that most of these networks have been designed through laborious trials of human experts, by tuning a large number of network hyperparameters such as operation type, the number of channels, connection, depth, etc. However, a network designed by human labor may not be optimal for the given resources.

In the case of image classification research fields, neural architecture search (NAS) methods have been proposed [19–26], which automatically find an optimal network to alleviate human labors [19–26]. As a pioneering study, Zoph *et al.* [19] proposed a controller network that generates a child network structure based on reinforcement learning (RL). The NAS trained the controller network by REINFORCE [28], which is a kind of policy gradient algorithm. But, this method took a tremendous amount of time to evaluate the candidate models because they trained the models from scratch. To reduce the time it takes to measure the accuracy, Liu *et al.* [20] proposed the PNAS, which used the sequential model-based optimization (SMBO) and learned a surrogate model to predict its performance directly. Also, ENAS [21] reduced the evaluation time by about a thousand times, by applying a weight sharing scheme. The ENAS constructed a large graph and regarded each model as a sub-graph of the main graph. In this way, child networks can share their parameters while being trained separately.

Another branch of NAS algorithms is the evolutionary-based methods [22–24], which pick a population of neural networks randomly. Then, they encode the network

structures as binary sequences and apply genetic modifications such as mutation and crossover to find better models. Additionally, NSGA-Net [23] used Bayesian optimization to get an advantage from its search history. Real *et al.* [24] introduced Amoeba-Net, which use an aging evolution algorithm to discard the earliest trained network. DARTS [25] and NAO [26] are also promising architectures that are approached differently from RL and evolutionary-based algorithm. DARTS optimized all parameters and connections in the neural architecture jointly with a continuous relaxation of the search space. NAO proposed a learnable embedding space of architectures and found the best model from it.

Recently, some researchers expanded the NAS to other domains such as object detection [27] (Auto-deeplab) and image SR [6] (MoreMNAS), [7] (FALSR). Specifically, they used a reinforced evolution algorithm and solved the image SR task as a multi-objective problem. However, these reinforced evolution algorithms need more than 1 GPU month to find optimal architectures. Furthermore, FALSR evaluated the network approximately because they did not use a complete training strategy.

To overcome these drawbacks, I take the idea of ENAS [21] as my baseline search framework. The ENAS consists of a controller and child networks, where the controller is composed of LSTM blocks to generate a child network sequence. I also use the REINFORCE [28] algorithm to train the controller in the direction of increasing PSNR of the SR results. As in the original ENAS for the classification problems, child networks share their parameters during the training and evaluation. In addition, I propose a complexity-based penalty to reduce the rewards for the networks that need a large number of parameters. I exclude redundant hierarchical information by predicting connections for local and global feature fusion through the controller.

SR is a type of regression that generally needs a deeper and more complex network than classification. Hence, in this dissertation, I search for a new SR architecture on the densely constructed search space. Specifically, I propose a Densely Connected Neural Architecture Search (DeCoNAS) method, which attempts to find optimal connections

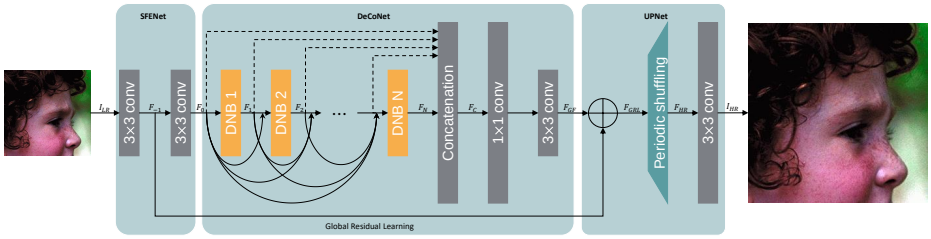



Figure 2.1: The architecture of the proposed DeCoNASNet system.

on the baseline of residual dense network (RDN) [17]. The proposed DeCoNAS search space consists of mix nodes for densely connected network blocks (DNB), local feature fusion, and global feature fusion. In addition to the convolution layer with $3 \times 3$ filters in the baseline network, dilated convolution [29] and depth separable convolution [30] are included in the search space for better performance. Experiments show that my DeCoNASNet performs better than human-crafted networks and the existing NAS-based SR network [6, 7]

my main contributions are summarized as follows:

1. A New NAS-Based SR: I propose a new NAS-based SR network design, named DeCoNAS, which searches for networks with higher performance by combining hierarchical and local information efficiently.

2. Complexity-Based Penalty: I design a complexity-based penalty and add it to the reward of the REINFORCE algorithm, which enables us to search for an efficient network that has high performance and fewer parameters.

3. Feature Fusion Layer Search: I also search for efficient feature fusion method. Instead of connecting all the global/local feature fusion layers, I design the connection to be predicted through the controller, which removes redundant hierarchical information and hence reduce network complexity.

## 2.2 Proposed Method

As a typical RL framework, the proposed architecture consists of two parts: a child network (denoted as DeCoNASNet) for reward measurement and a controller for network structure generation. Following ENAS [21], I try to save time by using parameter sharing when training a child network. Also, I regard the SR as a multi-objective task. That is, I design a complexity-based penalty to consider not only the PSNR but also the parameter complexity of the network for calculating the reward.

I use a two-layer LSTM network in my controller to generate the DeCoNASNet structure. Supposing that the child network **c** consists of $N$ blocks, $M$ mix nodes, and $K$ mix node operations, the controller sequence $\mathbf{S_c}$ for **c** is

$$\begin{aligned} \mathbf{S_c} &= \{\mathbf{S}_M, \mathbf{S}_F\}, \\ \mathbf{S}_M &= \{S_{i,j}^k\}, 0 < i \leq M, 0 \leq j < i, 0 \leq k < K, \\ \mathbf{S}_F &= \{{S_l}^0 : {S_l}^{M-1}, {S_g}^0 : {S_g}^{N-1}\}, \end{aligned} \tag{2.1}$$

where $\mathbf{S}_M$ is the sequence for mix node configuration, and $\mathbf{S}_F$ is the sequence for the feature fusion layer. $\mathbf{S}_F$ consists of two sequences, $\mathbf{S}_l$ and $\mathbf{S}_g$, which denote the sequence for local feature fusion and global feature fusion, respectively.

### 2.2.1 Overall structure of DeCoNASNet

DeCoNASNet consists of three parts as shown in Fig. 2.1, inspired by the RDN [17] architecture: shallow feature extractor network (SFENet), densely connected network (DeCoNet), and UPNet. The output of SFENet can be represented as

$$F_0 = H_{SFE2}(H_{SFE1}(I_{LR})), \tag{2.2}$$

where $H(\cdot)$ denotes the convolution operation. SFENet converts the input image $I_{LR}$ into a shallow feature $F_0$, which is used as the input to the DeCoNet. Then, the output

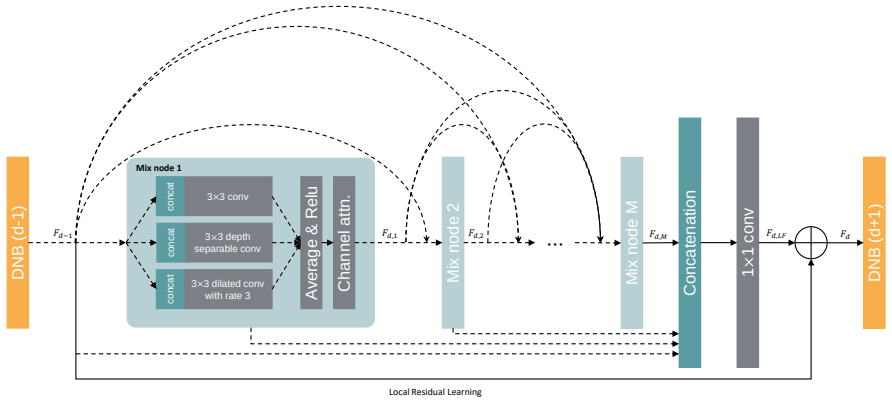



Figure 2.2: Detailed structure of the densely connected network blocks (DNB).

of the $d$-th DNB in DeCoNet, denoted by $F_d$ is expressed as

$$F_d = H_{DNB,d}(H_1(concat(F_0, F_1, \ldots, F_{d-1}))), \tag{2.3}$$

where $H_{DNB}(\cdot)$ denotes the DNB operation and $H_1(\cdot)$ is $1 \times 1$ convolution to match the input channels of DNBs. I will explain the operation $H_{DNB}(\cdot)$ in the next subsection. I omit $H_1(\cdot)$ and concatenation operation in Fig. 2.1 for simplicity. The global feature fusion layer follows after the $N$-th DNB to combine the information of the features of DNB. The output of the global feature fusion layer, denoted by $F_{GF}$ is described as

$$\begin{aligned} F_{GF} &= H_{GFF}(F_C), \\ F_C &= concat({S_g}^0 \cdot F_0, \ldots, {S_g}^{N-1} \cdot F_{N-1}, F_N), \end{aligned} \tag{2.4}$$

where $H_{GFF}(\cdot)$ denotes the $1 \times 1$ convolution and $3 \times 3$ convolution operation, ${S_g}^i$ denotes the output global feature fusion sequence of controller, and $concat(\cdot)$ denotes the concatenation of features. I omit $F_i$ when concatenating features if ${S_g}^i$ is zero. All of the ${S_g}^i$ equal to one if I do not use the feature fusion search strategy. The UPNet combines the output of DeCoNet and $F_{-1}$, which is the shallow feature from the SFENet. I use periodic shuffling operation and applied $3 \times 3$ convolution as in

ESPCN [14], to convert LR features to high-resolution images. I fix the SFENet and the UPNet, while I search for the DeCoNet. In all of my figures, I use dashed arrows to depict that the connection is to be searched.

### 2.2.2 Constructing the DNB

I apply the same operation $H_{DNB}(\cdot)$ through all the DNBs. Each DNB consists of M mix nodes as shown in Fig. 2.2, where there are three element candidates in the mix node:

- $3 \times 3$ 2D convolution,
- $3 \times 3$ depth separable convolution,
- $3 \times 3$ dilated convolution with rate 3.

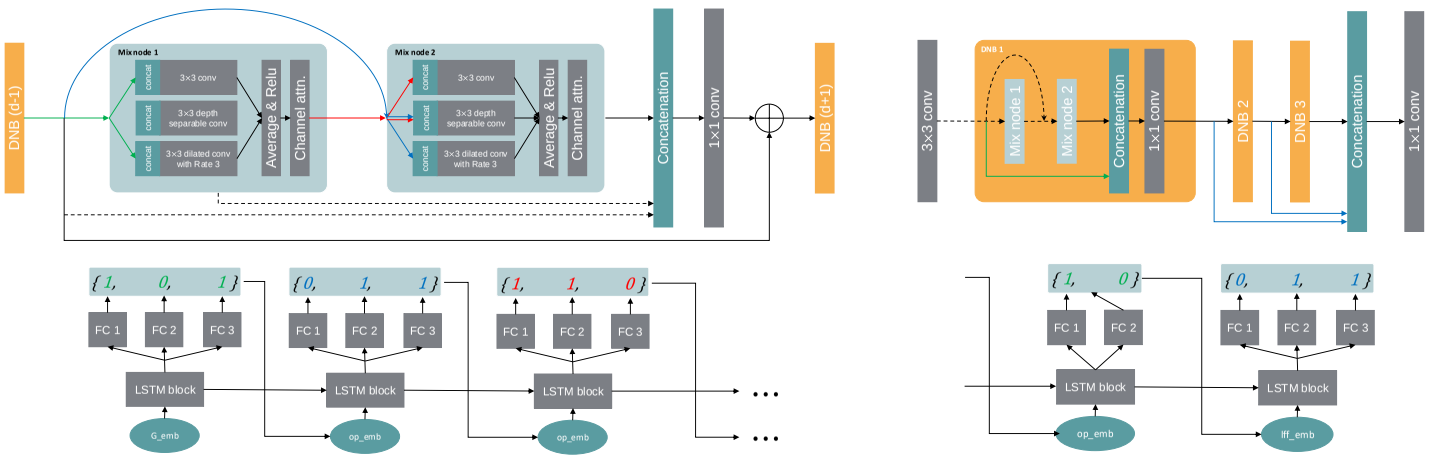
Also, $F_{d,m}$ over the arrow is the output of the $m$-th mix node in the $d$-th DNB with K candidate mix node operations, which are obtained as

$$F_{d,m} = \begin{cases} F_{d,m-1}, & \text{if } \mathbf{S}_{m,0:m-1}^{0:K-1} = \mathbf{0} \\ CA(Relu(average(\mathbf{F}_{inter}))), & \text{else} \end{cases} \tag{2.5}$$

where

$$\begin{aligned} \mathbf{F}_{inter} &= F_{d,m}^{0:K-1}, \\ F_{d,m}^{i} &= H_i(concat(S_{m,0}^{i} \cdot F_{d,0}, \ldots, S_{m,m-1}^{i} \cdot F_{d,m-1})) \end{aligned} \tag{2.6}$$

where $H_i(\cdot)$ denotes the $i$-th operation in $K$ operations, and $\mathbf{S}_{m,0:m-1}^{0:K-1}$ is the sequence for the $m$-th mix node configuration. Same as the Eq. (2.4), I omit $F_{d,m}^{i}$ or $F_{d,m}$ if $\mathbf{S}_{m,0:m-1}^{i} = \mathbf{0}$ or $S_{m,j}^{i} = 0$ in Eq. (2.6). $CA(\cdot)$ in Eq. (2.5) denotes channel attention network used in RCAN [34].




(a) (b)

Figure 2.3: The example structure of my Controller and DeCoNASNet structure with 2 mix nodes ($M = 2$) and 3 DNBs ($N = 3$). (a) is the example for mix nodes, and (b) is the example for feature fusion layers.

### 2.2.3 Constructing controller for the DeCoNASNet

#### Controller output of the mix node

I need $i \times K$ sequences to create the $i$-th mix node. Hence, my controller is composed of $\sum_{i=1}^{M} i = \frac{M(M+1)}{2}$ LSTM blocks, and $K$ fully connected layers are connected to each LSTM block. For example, As shown in Fig. 2.3(a), I use 3 LSTM blocks and 9 outputs to create the connections for two mix nodes.

#### Controller output of the feature fusion layer

There are two LSTM blocks for feature fusion layer search. These blocks are connected to the last LSTM block for mix node, in order to include the information about the mix node structure. Like the mix node, I connect $N$ and $M$ fully connected layers to two LSTM blocks, respectively. The output from each LSTM block denotes the connection between the mix node and the feature fusion layer. Fig. 2.3(b) shows an example connection of local/global feature fusion layer.

### 2.2.4 Training DeCoNAS and complexity-based penalty

Following ENAS [21], the DeCoNAS has two learnable parameters. The parameter of the controller is $\boldsymbol{\theta}$, and the parameter of the child network is $\mathbf{w}$. To learn $\boldsymbol{\theta}$ and $\mathbf{w}$ alternately, I use a two-step learning strategy. In the first step, I train $\mathbf{w}$ using training data. I use an RL scheme to train $\boldsymbol{\theta}$, with the reward signal consisting of peak signal to noise ratio (PSNR) and complexity-based penalty.

#### Training child network

As the first step to training DeCoNAS, I need to learn $\mathbf{w}$, which is the parameter of the child network, $\mathbf{c}$. This problem is defined as

$$\min_{\mathbf{w}} \mathop{\mathbb{E}}_{\mathbf{c}\sim\pi(\mathbf{c};\boldsymbol{\theta})}[L(\mathbf{c};\mathbf{w})]. \tag{2.7}$$

To optimize $\mathbf{w}$, I fix the controller's policy $\pi(\mathbf{c};\boldsymbol{\theta})$ and use Adam optimizer [35]. In this case, I use the L1 loss for $L(\mathbf{c};\mathbf{w})$, calculated on training data and model $\mathbf{c}$ generated by $\pi(\mathbf{c};\boldsymbol{\theta})$. Gradient of $\mathbb{E}_{\mathbf{c}\sim\pi(\mathbf{c};\boldsymbol{\theta})}[L(\mathbf{c};\mathbf{w})]$ is calculated by Monte Carlo estimate

$$\bigtriangledown_{\mathbf{w}} \mathop{\mathbb{E}}_{\mathbf{c}\sim\pi(\mathbf{c};\boldsymbol{\theta})}[L(\mathbf{c};\mathbf{w})] \approx \frac{1}{M}\sum_{i=1}^{M} \bigtriangledown_{\mathbf{w}} L(\mathbf{c}_i;\mathbf{w}), \tag{2.8}$$

where $\mathbf{c}_i$'s are sampled by the controller's policy $\pi(\mathbf{c};\boldsymbol{\theta})$. As mentioned in ENAS [21], $\mathbf{w}$ can be optimized by calculating the gradient for only one model $\mathbf{c}$ generated by $\pi(\mathbf{c};\boldsymbol{\theta})$ for each mini-batch.

**Training controller with performance reward and complexity-based penalty**

In the second step, I need to train $\boldsymbol{\theta}$, which is the controller's parameter. This problem is to maximize the expected reward as

$$\max_{\boldsymbol{\theta}} E_{P(a_{1:T};\boldsymbol{\theta})}[R], \tag{2.9}$$

where $a_{1:T}$ is the controller output for the child network $\mathbf{c}$, which follows the distribution of $\pi(\mathbf{c};\boldsymbol{\theta})$. I compute the gradient of the problem by using the approximation of gradients in REINFORCE [28] as

$$\bigtriangledown_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{P(a_{1:T};\boldsymbol{\theta})}[R] = \sum_{t=1}^{T}[\bigtriangledown_{\boldsymbol{\theta}} \log P(a_t|a_{t-1:1};\boldsymbol{\theta})(R-b)] \tag{2.10}$$

where $b$ is the baseline for reducing the variance, which is the moving average of the reward. While ENAS used classification accuracy for $R$, I calculate it differently as

$$R = p(\mathbf{c},\mathbf{w}) - \alpha * cb(\mathbf{c}), \tag{2.11}$$

where $p(\mathbf{c},\mathbf{w})$ is the PSNR of model $\mathbf{c}$. I calculate the PSNR using the validation set rather than the training set to prevent overfitting. Also, $cb(\mathbf{c})$ is the complexity-based

penalty, calculated as

$$cb(\mathbf{c}) = \frac{n_m}{n_{cm}}, \tag{2.12}$$

where $n_m$ denotes the number of parameters in the generated model $\mathbf{c}$ and $n_{cm}$ indicates the number of parameters in the most complex model in the search space. I also multiply $\alpha$ by $cb(\mathbf{c})$, allowing the user to set a trade-off between the performance and model complexity. Finally, I use Adam optimizer [35] to maximize the reward.

## 2.3 Experimental results

### 2.3.1 Settings

#### Datasets and metrics

I use DIV2K dataset [36] for the training, which has been widely used for training image restoration networks. The DIV2K contains 800 training images, 100 validation images, and 100 test images. I use all images in the training set to train my DeCoNASNet, and use all of the validation images when calculating the reward and training the controller. Experiments are conducted on four benchmark datasets, Set5 [37], Set14 [38], B100 [39], and Urban100 [40], where I compute PSNR and SSIM [41] on the Y channel.

#### Implemenation details

My proposed DeCoNASNet has 4 DNBs, and each DNB has 4 mix nodes. The output channel of SFENet and DNB are both 64. The $3 \times 3$ convolution layer in UPNet also has 64 output channels, and I conduct periodic shuffling on the feature maps. The final convolution layer has $3 \times 3$ filters and three output channels to restore the high-resolution images.

The LSTM block in the controller is made of two stacked LSTM layers with 64 hidden states. Fully connected layers for one operation in each LSTM block are sharing

their parameters. I tie the LSTM outputs with word embeddings [42] to make the input of the next LSTM block.

#### Training setting

In the search phase, I need to train controller and DeCoNASNet together. I use variance scaled initialization [43] with 0.02 scaling value for DeCoNASNet parameter $\mathbf{w}$ and controller parameter $\boldsymbol{\theta}$. To train the controller, I apply 100 iterations for one epoch, and the learning rate is fixed to $3 \times 10^{-4}$.

For training the DeCoNASNet, I randomly extract 16 LR patches of size $64 \times 64$ from the DIV2K training image as the input to the network. After extracting the patches, I apply horizontal flip and 90°, 180°, 270° rotations to each patch randomly for data augmentation. I conduct 1,000 backpropagation for an epoch, where Adam optimizer [35] is used for updating parameters. The learning rate is initialized to $10^{-4}$ and decreased by half for every $5 \times 10^5$ iterations (50 epochs), and 200 epochs are conducted for the search phase. I sample 100 candidate structures by the trained controller and choose the best architecture as my DeCoNASNet structure. After choosing the best architectures, I train the DeCoNASNet for 1,000 epochs. The learning rate is initialized to $10^{-4}$ and decreases by half for every 200 epochs. The other settings are the same as the search phase.

### 2.3.2 Results

#### DeCoNAS search result

I use 16 DNBs ($N = 16$) and 8 mix nodes in each DNB ($M = 8$) to verify the effect of feature fusion strategy and complexity-based penalty. Table 2.1 and Fig. 2.4 show the performance of total 400 DeCoNAS structures in four settings (100 structures each). The baseline setting (denoted as FF0_CB0), which omitted the feature fusion search (FF) and complexity-based penalty (CB), shows the best performance with 23.2 M parameters. I add one of CB or FF to FF0_CB1 and FF1_CB0. From the results, I can

Table 2.1: Performance comparison between search settings.

| Search setting | | CB0_FF0 | CB0_FF1 | CB1_FF0 | CB1_FF1 |
|---|---|---|---|---|---|
| Best | PSNR | **35.919** | 35.602 | 35.894 | 35.436 |
| | Parameters | 22.9 M | 25.4 M | **18.0 M** | 25.9 M |
| Mean | PSNR | 35.294 | 35.168 | **35.324** | 35.025 |
| | Parameters | 22.5 M | 28.4 M | **17.1 M** | 24.9M |

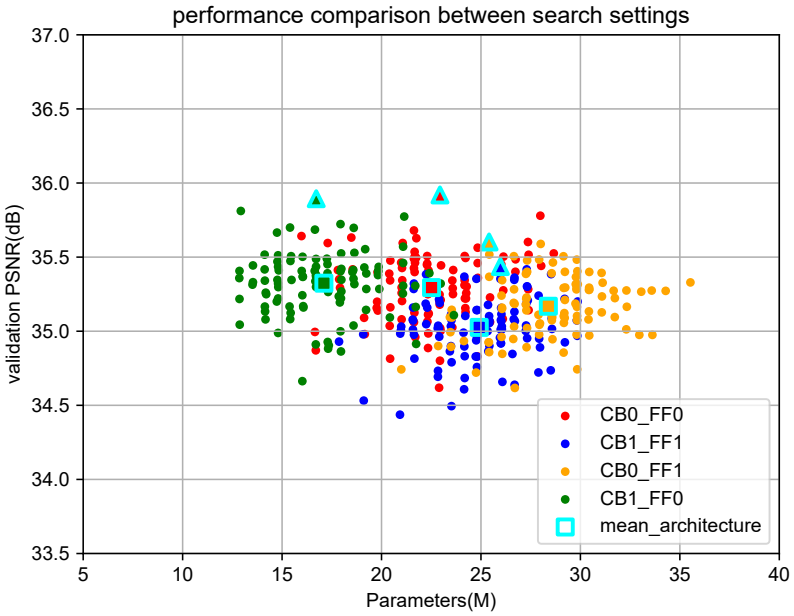



Figure 2.4: Scatter plot of 100 samples for each search setting.

also see that FF0_CB1 achieves almost the same performance as the baseline search strategy, with 20% less parameters, and FF1_CB0 has slightly lower performance with large parameters. Finally, I apply both strategies, resulting in FF1_CB1, which is shown to have fewer parameters than the FF1_CB0 model, but yields inferior performance than the others. Further discussions about CB and FF strategies are at the ablation study section.

I use 4 DNBs ($N = 4$) and 4 mix nodes ($M = 4$) in each DNB to make De-

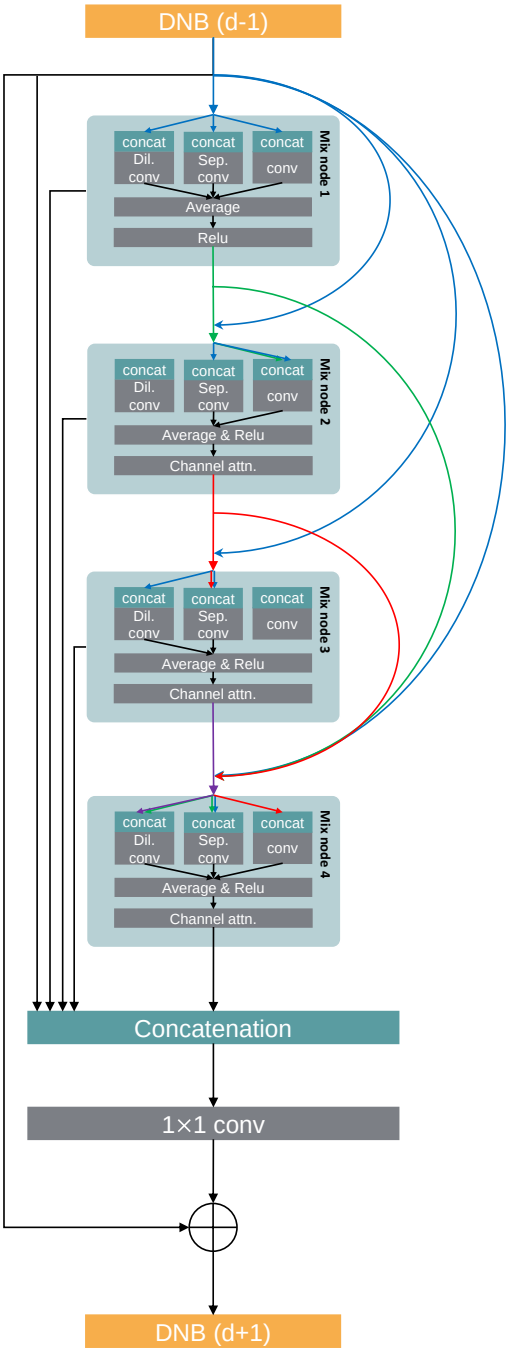



Figure 2.5: My DeCoNASNet model found by controller.

Table 2.2: Public benchmark test results (PSNR/SSIM) for $\times 2$ SR. The red color means the best performance and the blue means the second best. The "Design time" at the last column indicates the times taken by the NAS approaches.

| Model | Params | Set 5 | Set 14 | B100 | Uban100 | Design time |
|---|---|---|---|---|---|---|
| Bicubic | - | 33.66 / 0.9299 | 30.24 / 0.8688 | 29.56 / 0.8431 | 26.88 / 0.8403 | - |
| SRCNN [1] | 57K | 36.66 / 0.9542 | 32.45 / 0.9067 | 31.36 / 0.8879 | 29.50 / 0.8946 | - |
| VDSR [2] | 665K | 37.53 / 0.9587 | 33.03 / 0.9124 | 31.90 / 0.8960 | 30.76 / 0.9140 | - |
| LapSRN [3] | 813K | 37.52 / 0.9591 | 33.08 / 0.9130 | 31.80 / 0.8950 | 30.41 / 0.9101 | - |
| MemNet [4] | 677K | 37.78 / 0.9597 | 33.28 / 0.9142 | 32.08 / 0.8978 | 31.31 / 0.9195 | - |
| SelNet [33] | 970K | 37.89 / 0.9598 | 33.61 / 0.9160 | 32.08 / 0.8984 | - / - | - |
| CARN [5] | 1,582K | 37.76 / 0.9590 | 33.52 / 0.9166 | 32.09 / 0.8978 | 31.92 / 0.9256 | - |
| MoreMNAS-A [6] | 1,039K | 37.63 / 0.9584 | 33.23 / 0.9138 | 31.95 / 0.8961 | 31.24 / 0.9187 | 56 GPU days |
| FALSR-A [7] | 1,021K | 37.82 / 0.9595 | 33.55 / 0.9168 | 32.12 / 0.8987 | 31.93 / 0.9256 | 24 GPU days |
| DeCoNASNet (mine) | 1,713K | 37.96 / 0.9594 | 33.63 / 0.9175 | 32.15 / 0.8986 | 32.03 / 0.9265 | 12 GPU hours |

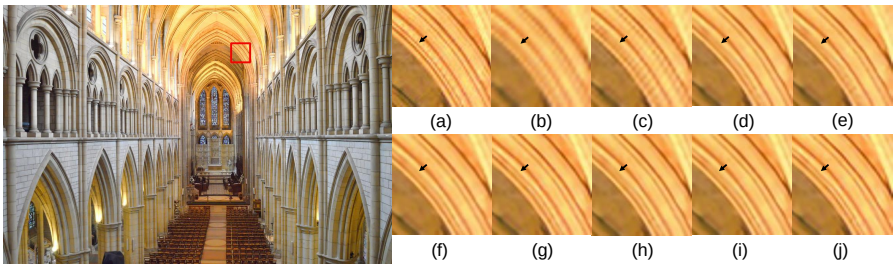



Figure 2.6: Qualitative comparison of the conventional methods and mine. (a) HR image, (b) bicubic LR image, (c) SRCNN [1], (d) VDSR [2], (e) LapSRN [3], (f) MemNet [4], (g) CARN [5], (h) MoreMNAS [6], (i) FALSR [7], (j) DeCoNASNet (mine).

CoNASNet structure. I choose the best architecture which belongs to FF0_CB1 setting. Fig. 2.5 shows the DNB structure of DeCoNASNet:{7, 6, 4, 3, 0, 2, 2, 3, 4, 1}. The local/global feature fusion connections are all connected because I do not use the feature fusion search strategy. I regard three outputs of each controller LSTM block as the binary number and convert it to a decimal number for simplicity. For example, four in the sequence means {1, 0, 0} and six is {1, 1, 0}. It takes about 12 hours by 1 Titan XP GPU to search for the DeCoNASNet structure, which is far less than other NAS-based methods.

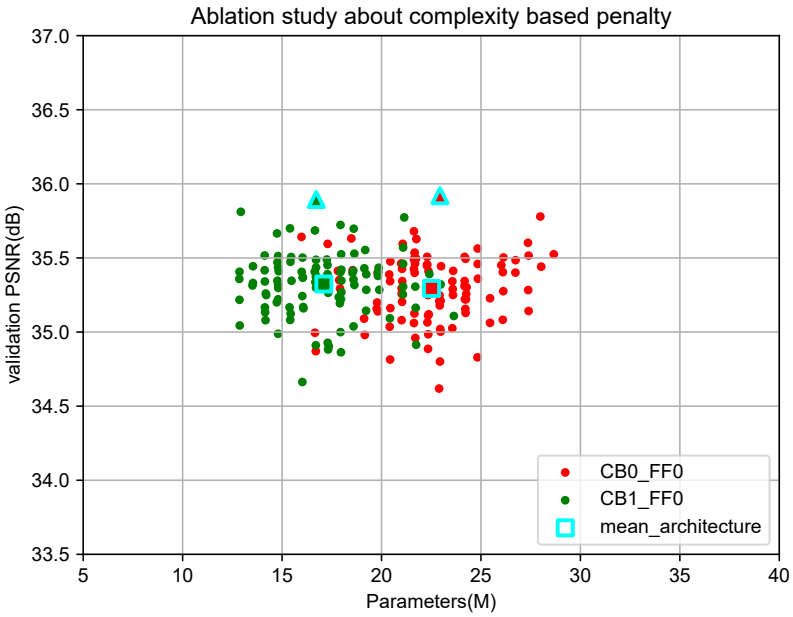



Figure 2.7: Ablation analysis about effect of the complexity-based penalty.

**Comparison with state-of-the-art methods**

I compare my DeCoNASNet with six lightweight networks (SRCNN [1], VDSR [2], MemNet [4], LapSRN [3], SelNet [33], CARN [5]) and two NAS-based methods (MoreMNAS [6], FALSR [7]). The results are shown in Table 2.2, where I can see that DeCoNASNet outperforms other hand-crafted lightweight models and existing NAS-based ones while using somewhat more parameters, still within 2M. The most important advantage of my method is that it finds the optimal structure within 16 hours, which is $\times 50$ faster than other NAS-based methods. I compare the visual results of my model with the others (SRCNNN [1], VDSR [2], LapSRN [3], MemNet [4], CARN [5], MoreMNAS [6], FALSR [7]) in Fig. 2.6. I find that DeCoNASNet successfully restores the details in images. Specifically, DeCoNASNet and CARN restore the double curves at the black arrow, while others do not.

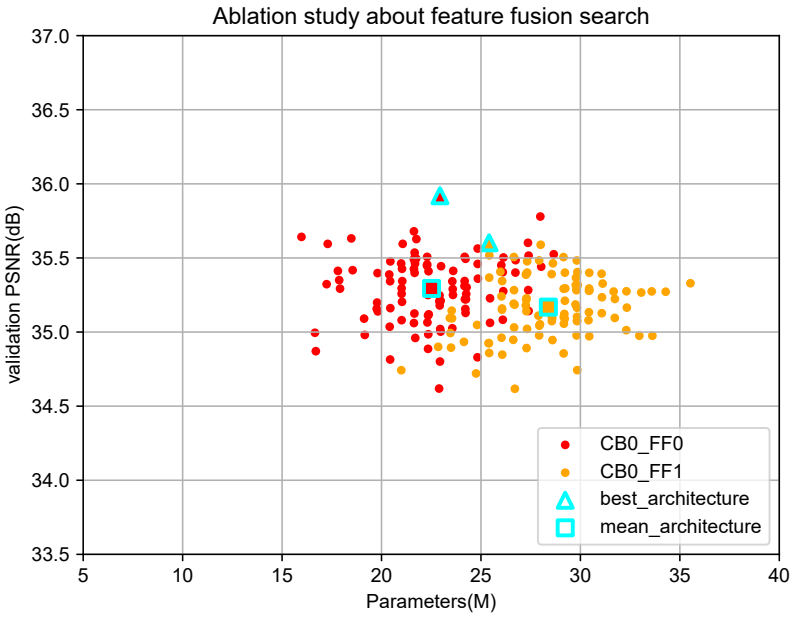



Figure 2.8: Ablation analysis about the feature fusion search strategy.

### 2.3.3 Ablation study

Fig. 2.7 shows a comparison between different complexity-based penalty coefficients, $\alpha$. I conduct two experiments in the DeCoNAS search space ($N = 16$, $M = 8$) to analyze the effect of the complexity-based penalty. The baseline strategy is denoted as CB0_FF0, and the other strategy using CB coefficient $\alpha = 2$ is described as CB1_FF0. The parameters of the searched model are decreased, but the performance difference is small when I change the CB strategy. Hence, I can validate that the complexity-based penalty efficiently controls the trade-off between the performance and the number of parameters. Users can choose $\alpha$ to find models that fit their purpose.

I also show the effect of feature fusion search (FF) strategy in Fig. 2.8. The red and orange dots in the figure denote CB0_FF0 and CB0_FF1, respectively. It can be seen that the FF strategy tends to find more complex models than no FF strategy. This is mainly because the controller tries to compensate for the information loss from the

feature fusion layer disconnection. I can verify that the connections to the global/local feature fusion layer are more important than the connections in the mix node.

## 2.4 Summary

I have proposed an RL-based neural architecture search algorithm for image SR, named as DeCoNAS. It is shown that the proposed method can find a promising lightweight SR network (DeCoNASNet) within 16 hours, which is a lot faster than other NAS-based algorithms. I have also proposed a feature fusion search strategy in the proposed searching scheme, which verified the importance of global/local fusion structure for the SR. Moreover, the complexity-based penalty to the reward could reduce the network complexity, which enabled lightweight network architecture. Experiments show that the resulted DeCoNASNet yields higher performance in terms of PSNR vs. complexity among the recent handcrafted lightweight SR networks and other NAS-based ones.

# Chapter 3

# Multi-Branch Neural Architecture Search for Lightweight Image Super-resolution

## 3.1 Introduction

Single image super-resolution (SISR) is a task that restores a high-resolution (HR) image from a single low-resolution (LR) observation. It is widely used as a preprocessing step of various tasks such as medical image analysis [8], security image processing [10], satellite image recognition [9], etc. Most of the recent researches adopt learning-based methods that use LR-HR image pairs for training [1–5, 13–18], which generally show better performance than the classic interpolation-based [11] or reconstruction-based [12] methods.

Most earlier learning-based SR methods used single-branch neural networks for their simplicity and straightforwardness. However, when the single branch is deepened to increase the performance, there can be a gradient vanishing problem, and the resulting network needs too many parameters. Thus, instead of using the single branch network, some methods exploited multi-branch networks for extracting multi-scale features from the LR input [15, 18, 31, 32], thereby achieving better performances with fewer parameters. But due to the increased complexity of the network structure, it

needs many trials and errors to find the optimal connection between the elements manually. Based on this, in this dissertation, I propose an automated multi-branch SISR network design based on the neural architecture search (NAS) scheme [19], unlike the conventional manual design of single-branch or multi-branch networks. I include the multi-branch networks to expand the search space and propose a new NAS-based SISR network design while existing search methods attempted to find the optimal connection within the single-branch networks.

Neural architecture search (NAS) algorithm has been developed for the purpose of reducing the effort put into designing the neural architecture of certain tasks [19–26, 44]. They focus on the image classification task and try to find promising network automatically by adopting reinforcement learning, evolutionary algorithm or gradient descent method.

Recently, researchers have expanded the NAS to other tasks such as image restoration [6, 7, 45, 46] (MoreMNAS, FALSR, HNAS, Improved DARTS), and object detection [27]. For the SISR, FALSR and MoreMNAS used a reinforced evolution algorithm and solved the image SR task as a multi-objective problem. However, the reinforced evolution method took a tremendous amount of time to derive an optimal network. Additionally, FALSR did not use a complete training scheme, but they measured the performance of the network approximately.

To alleviate these problems in my application, I adopt the weight-sharing scheme of ENAS [21] as my baseline search algorith because it is known to provide faster design time than its predecesors. As in the original ENAS for the classification problems, I configure a controller and a child network in the search process. The controller generates a sequence for a child network, and a child network is constructed by the generated controller sequence. REINFORCE algorithm is used to train the controller network to generate a better child network. For the SISR task, The reward signal in REINFORCE is the PSNR between the generated child network's output and the ground-truth. I share the parameters of each child network during the search phase. In addition, I propose a

complexity-based penalty to reduce the reward from the network that needs a huge parameter. By applying the complexity-based penalty, the controller tends to recommend powerful but lightweight networks.

Image super-resolution is a kind of regression task that generally requires a more precise and complex network than a classification task. For this reason, I search for a new SR architecture on a multi-branch search space as stated above. To be specific, I propose a Multi-Branch Neural Architecture Search (MBNAS) algorithm, which tries to find optimal connections of multi-scale features. The MBNAS search space consists of partially shared nodes (PSN) for multi-scale block, local feature fusion layer, and global feature fusion layer. The PSNs share their parameters with different network branches to transmit information efficiently with fewer parameters. For simplicity, I use only $3 \times 3$ convolution and $3 \times 3$ dilated convolutions [29] as basic building blocks, and let the search algorithm find optimal connections. Still, I obtain an efficient architecture as a result of the search algorithm, which is validated by extensive experiments. The experimental results show that my network obtained by the MBNAS, named as MBNASNet, performs comparably to human-crafted networks and the existing NAS-based SR networks [6, 7, 45].

My main contributions are summarized as follows:

1. New NAS-based SR: I propose a new NAS-based SR network design method, named MBNAS, which searches for networks with higher performance by combining multi-scale information efficiently. The resulting SR network is the MBNASNet.

2. Complexity-based penalty: I propose a complexity-based penalty and add it to the reward signal of the REINFORCE algorithm. This enables us to search for an efficient network that has high performance with a lightweight structure.

3. Multi-branch structure for multi-scale feature extraction: I construct the network with a multi-branch structure, and each branch learns how to restore patterns of

different scales.

4. Partially shared node (PSN): I partially share the parameters of branches to connect each other's information and construct a lightweight structure. The partially shared structure efficiently reduces the searched network's parameter without performance degradation.

I presented a preliminary work of NAS-based image super-resolution with a single-branch network in [47], called DeCoNASNet. The major difference of this work from my previous version is that I propose an expanded search space for NAS to capture multi-scale information, which brings a significant performance gain with reduced parameters. For this, I modify the algorithm to include the multi-braches into the search space. Also, I provide detailed analysis and explanations of the search process and results, and exhibit more experimental results, including the results on higher rate SR.

The rest of this chapter is organized as follows. Section 2 summarizes related works on the single image super-resolution and neural architecture search methods. In section 3, I explain my proposed search method for SISR. Section 4 includes the details about my implementation settings and dataset configurations, followed by experiment results. I discuss my main contributions and conduct ablation experiments in section 5. Finally, I provide a summary and concluding remarks in section 6.

## 3.2 Related Work

### 3.2.1 Single image super-resolution

A number of methods have been proposed for learning the mapping function from LR images to the appropriate HR counterparts [1–5, 13–18]. Dong *et al.* proposed SR-CNN [1], which is the first deep learning structure for the SISR. It used three layers of convolutional neural networks (CNNs) and outperformed non-learning-based conventional methods by a large margin. FRCNN [13] and ESPCN [14] used specific struc-

tures to reduce the computational cost of deep neural networks in the SISR networks. They proposed deconvolution layers and sub-pixel convolution layers to upsample LR features to an HR image. VDSR [2] used residual learning and gradient clipping strategy to increase the depth and thus the performance. Lim *et al.* [15] introduced residual blocks with extensive features (EDSR) and multi-scale structure (MDSR) to improve the performance further. MemNet [4], MSRN [18], and DenseSR [16] proposed memory block, multi-scale residual block, and dense block, respectively, for a better SR restoration. SelNet [33] improved the performance by replacing the Relu operation with the selection unit. Zhang *et al.* proposed residual dense block and dense feature fusion algorithm in RDN [17] to extract abundant information from the input image. RCAN [34] proposed a channel attention scheme that improved the representational ability of the neural network.

### 3.2.2 Neural architecture search

In designing a deep network, I should select a considerable number of network configurations such as connection, operation type, the number of feature channels, depth, etc. Researchers have designed their structures through a large number of trials to achieve a competent performance. However, it is a tedious task and difficult to find an optimal system for a given task. The NAS algorithms have been proposed to alleviate this burden, especially in the case of image classification researches [19–26].

As the first study of NAS, Zoph *et al.* [19] proposed a reinforcement learning (RL) based algorithm. They configured a controller network to generate a child network and trained it by REINFORCE [28], which is a kind of policy gradient algorithm. The performance of the child network was used as a reward signal of the controller network, where the child network was trained from scratch. Therefore, it took a huge amount of time to get a reward signal from the child network. To reduce the time to measure the performance, PNAS by Liu *et al.* [20] used the sequential model-based optimization (SMBO) with a surrogate model which predicts its performance instantly.

On the other hand, Pham *et al.* [21] proposed ENAS that constructs a weight sharing child network to reduce the reward calculation time. This method configured a large graph and regarded each child network as a sub-graph. The parameters of the child network were shared in the search phase by storing their weights in the main graph.

Evolutionary methods [22–24] are another trend of the NAS algorithm. They pick a population of architectures randomly at first and then encode these networks as binary codes. Genetic modifications such as crossover or mutations are applied to the sequence, suggesting a better structure. Lu *et al.* [23] proposed another method that takes advantage of search history by using a Bayesian optimization algorithm. AmoebaNet [24] applies an aging evolution method to NAS to discard the earliest trained network.

DARTS [25], SGAS [48], NAO [26] and CSA-NAS [44] proposed different approaches from RL and evolutionary methods. Specifically, DARTS applies continuous relaxation to the neural architecture's connections for optimizing the connections and parameters simultaneously. SGAS applies a greedy operation selection method to the DARTS and obtains the best architecture without retraining. NAO projects the encoded sequence to the learnable embedding space of structures and recommends the best architecture as a result. CSA-NAS adopts a binary crow search algorithm to find the optimal architecture.

Regarding the search space design, neural architecture search methods can be categorized into two groups: methods dealing with (1) flat search space or (2) cell-based search space. The methods with flat search space [19, 21–23] aim to find the optimal setting for the number of channels (width), number of layers (depth), types of operations (convolution or max pooling) for the whole structure, while cell-based algorithms [20, 21, 23–26] try to find a structure of the cell before stacking them to form the final architecture. The cell-based search space design is inspired by the split-transform-merge strategy used in Inception block [49], hence it can approximate the optimal solution for a given task.

Unlike the above algorithms, CSNAS [50], UnNAS [51], and SSNAS [52] discard supervised settings which suffer from the high cost of data labeling. CSNAS and SSNAS adopt a self-supervised setting, and UnNAS applies unsupervised learning to search for promising architectures with unlabeled data. Recently, researchers are also trying to overcome the reproduction challenge and fairly compare search methods by proposing benchmarks for the NAS and providing some important principles for scientific research in the community [53–55].

There have been many NAS methods as stated above, among which I choose ENAS as my SR design baseline for its fast design time and also for including the network complexity in the design constraints. Regarding the design time, DARTS [25], FBNet [56], and FBNetV2 [57] also provide fast design time for practical use. But, I choose ENAS as my SR design baseline because I can easily include the complexity constraint into consideration within the ENAS framework. Specifically, as the ENAS is based on the REINFORCE, I modify the reward signal of the REINFORCE to consider the network complexity as well as the SR performance.

### 3.2.3 Image super-resolution with neural architecture search

Some researchers recently adopted NAS methods to design image super-resolution CNNs [6, 7, 45]. MoreMNAS [6] adopted multi-objective genetic algorithm NSGA-II [58] for the model generation and proposed a reinforced mutation method. FALSR [7] used a hybrid controller instead of a reinforced controller and proposed an elastic search space for macro and micro search. The search space complexity of both methods is $9.6 \times 10^{15}$. HNAS [45] adopted a hierarchical search algorithm with reinforcement learning to simultaneously find promising cell structure and upsampling layer positions. They also considered the computational cost (FLOPS) to meet the requirements about resources constraint. HNAS searches the network from $1.03 \times 10^{9}$ candidate networks.

Regarding the architecture and the search space thereof, these previous NAS-based

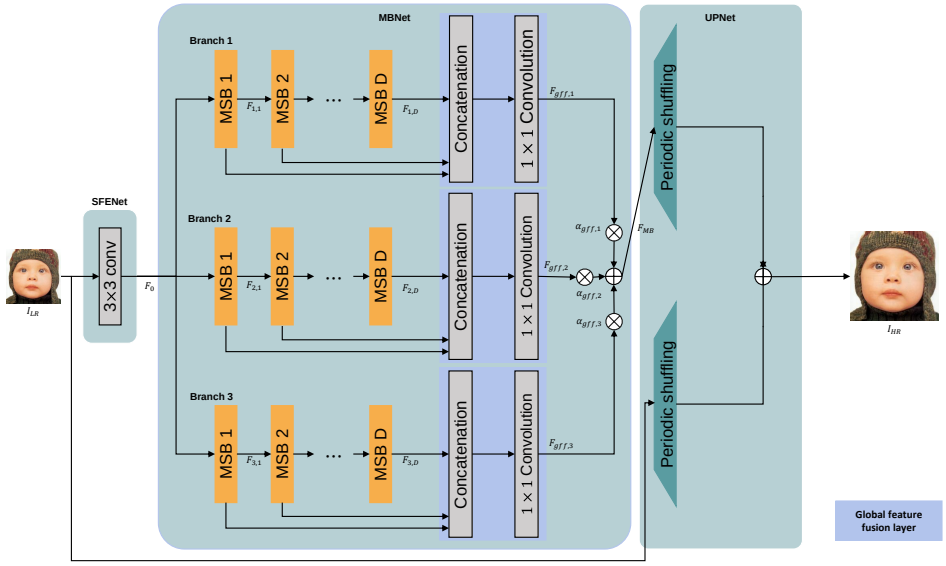
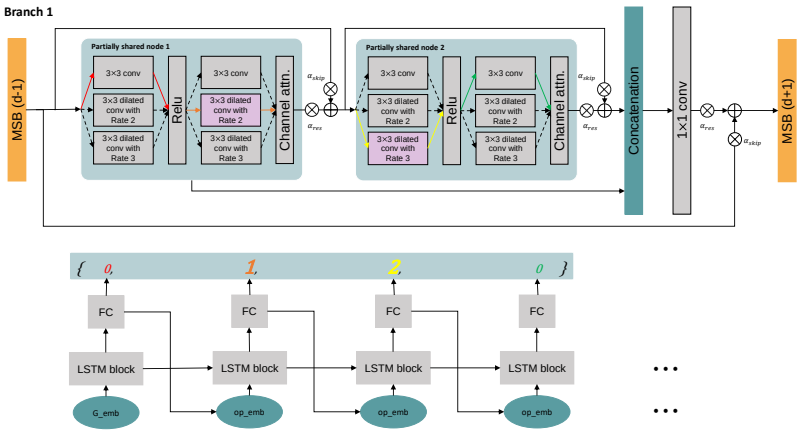
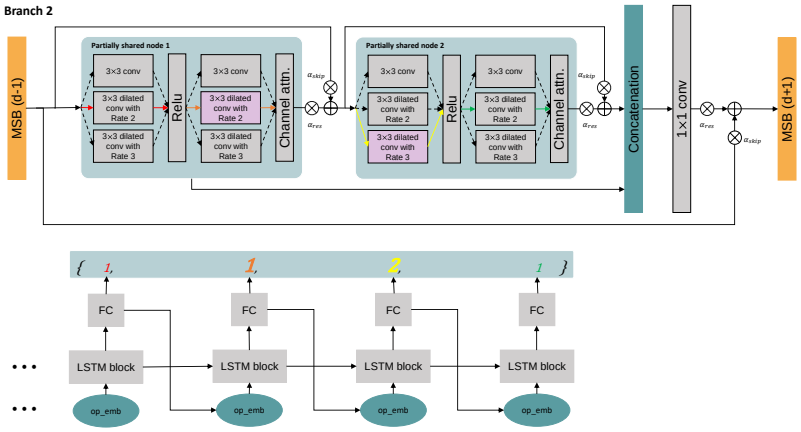



Figure 3.1: The overall structure of MBNASNet. It consists of a shallow feature extraction network (SFENet), a multi-branch network (MBNet), and an upscaling network (UPNet). The result of each branch is combined and upsampled by the periodic shuffling layer. The MSB (multi-scale block) is a basic building block, detailed in Fig. 3.2(a).

methods prepare basic building blocks, which consist of convolutional layers, ReLu, etc., in cascade. Then, they let the NAS algorithm determine the number of layers and connections inside the cells. Meanwhile, I prepare a sophisticated architecture to have expanded search space, *i.e.*, a structure with more different functional elements to connect. Specifically, I prepare several branches of building blocks, consisting of multi-rate dilated convolutions, ReLu, and attention, and let the NAS algorithm find the connections among the various-scale convolutions. By expanding search space through the multi-branch of dilated convolutions, I can exploit multi-scale features for better SR reconstruction than conventional single-branch architecture.




(a)




(b)

Figure 3.2: The upper part of (a) shows details of my MSB, and the lower part is illustrating that a controller determines the connections inside the MSBs of branch 1 according to the controller sequence (outputs of FC layers), with an example that there are two partially shared nodes (PSNs) $(M = 2)$ and two branches $(B = 2)$. (b) shows the example for branch 2, where the elements inside the MSB are differently connected than the above case according to the corresponding controller. Two branches share the parameters of the light purple box. The dashed arrows and colored arrows mean that these connections are to be searched.

## 3.3 Method

### 3.3.1 Overview of the Proposed MBNAS

My MBNASNet (a child network) is shown in Fig. 3.1, whose components (MSBs) are designed by a controller in Fig. 3.2, according to the MBNAS algorithm of Fig. 3.3. The automated design cycle in Fig. 3.3 illustrates that the controller is trained to generate a potent network, and the child network is trained to get the performance, which is used to calculate the reward signal.

Fig. 3.1 shows the overview of MBNASNet, which consists of a shallow feature extraction network (SFENet), an upscaling network (UPNet), and a multi-branch network (MBNet). The MBNet is designed by the NAS, which consists of several branches. The MSB (multi-scale block) in the figure is the basic building block detailed in Fig. 3.2. I extract a shallow feature by the SFENet that is fed to each branch. The partially shared parameters in each branch extract the multi-scale features with different receptive fields. Results from each branch are combined and upsampled by pixelshuffle layers [14] to create HR residual information. Finally, the residual information is added to the upsampled LR input to make the final HR result.

Fig. 3.2 shows the details of MSB and illustrates their internal connections according to sequences from the controller. In each of Fig. 3.2(a) and (b), the upper part shows a branch of MBNASNet in Fig. 3.1, where three consecutive MSBs are shown. The central part details the structure of the $d$-th MSB, and the left and right are the $(d-1)$-th and $(d+1)$-th MSBs. The lower part shows the controller that outputs a sequence to determine the internal connections of the MSB. Fig. 3.2(a) and (b) show different examples of the output sequences from the controller and the corresponding connections inside the MSBs.

I use Long Short Term Memory (LSTM) [59] to create the controller, where the parameters are updated by REINFORCE algorithm. While conventional RL methods calculate the reward signal of REINFORCE as the performance of validation sets, I

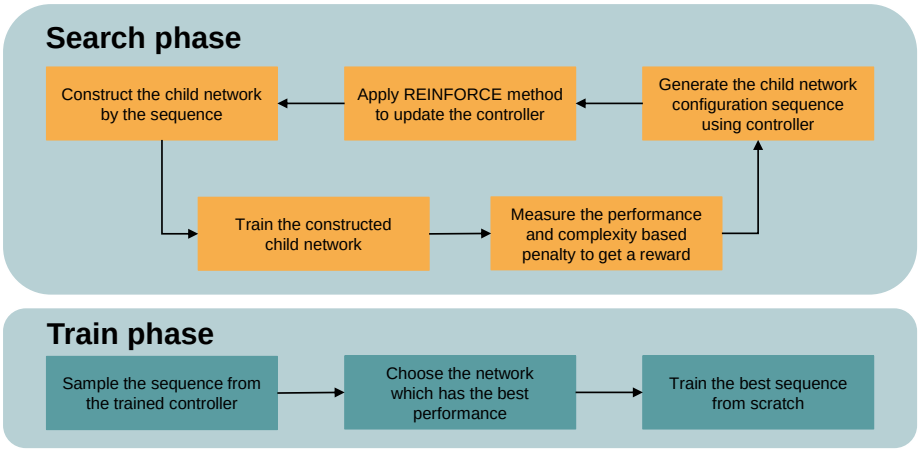



Figure 3.3: The overview of the search cycle and training. In the search phase, the controller and constructed child network are trained alternatively. In the training phase, the searched final architecture is trained from scratch.

consider both performance and network complexity. For this, I design a complexity-based penalty and add it to the reward signal to find a more efficient architecture. The details of the controller, MBNASNet, and design procedure are explained in the rest of this Section.

### 3.3.2 Controller and complexity-based penalty

#### Controller configuration

I use a two-layer LSTM as my controller as shown in the lower part of Fig. 3.2. It generates a sequence for creating a child network at the end of the fully connected layer (FC). The output sequence $\mathbf{S}_c$ for a child network $c$ is defined as

$$\begin{aligned} \mathbf{S}_c &= \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_b, \ldots, \mathbf{s}_B\}, \\ \mathbf{s}_b &= \{(s_b)_{m,n}\}, 0 < m \leq M,\ 0 \leq n < N, \end{aligned} \tag{3.1}$$

in the case that the child network consists of $B$ branches, $M$ PSNs in one multi-scale block (MSB), and each node has $N$ layers. $\mathbf{S}_c$ consist of $B$ sequences, and each $\mathbf{s}_b$

denotes the sequence of the $b$-th branch structure. I need $N$ sequences to create the $m$-th PSN for one branch. As a result, my controller consists of $M \times N \times B$ LSTM blocks, where each block is followed by an FC layer. The FC layer has $K$ outputs, where $K$ is the candidate operations of my network. The example sequence and the constructed block architecture are shown in Fig. 3.2, which generate eight outputs for a two-branch structure ($B = 2$) with two PSNs ($M = 2$) that have two layers ($N = 2$).

In my search space, the total number of possible directed acyclic graphs (DAGs) is $|K|^{B \times M \times N}$. The set of all possible neural architecture is enormously expanded by a factor of $|K|^{M \times N}$ when increasing the number of branches. The search space is also expanded if I increase the number of PSNs or their layers. Hence, to limit the number of possible architectures to a manageable size, I choose $B = 3$, $M = 2$ and $N = 2$ in my MBNASNet. Because I have three candidate operations ($|K| = 3$), as will be addressed in Sec. III-D, the possible set of the architecture is $5.3 \times 10^5$. Finally, to ensure that the number of parameters is less than 2M, I construct my MBNASNet with four multi-scale blocks ($D = 4$).

**Complexity-based penalty**

The REINFORCE algorithm uses a reward signal to train the parameters of the controller. While ENAS uses only a task performance as the reward signal, I modify the reward signal to find a more powerful and lightweight architecture, as stated in overview section. Specifically, I propose a complexity-based penalty to penalize a structure with large parameters, and define a reward signal $R$ as

$$R = p(c; \mathbf{w}) - \lambda \times cb(c), \tag{3.2}$$

where $p(c; \mathbf{w})$ is the PSNR of model $c$ and $\mathbf{w}$ is the parameters of a child network. The complexity-based penalty, $cb(c)$ is defined as

$$cb(\mathbf{c}) = \frac{n_c}{n_{max}}, \tag{3.3}$$

where $n_{max}$ denotes the number of the model's parameters, which uses all candidates in the search space, and $n_c$ is the number of parameters of the designed child network. To set a trade-off between the parameters and the performance, I multiply $\lambda$ to the complexity-based penalty.

### 3.3.3 MBNASNet

As shown in Fig. 3.1, I first extract a shallow feature $F_0$ from an input low-resolution image ($I_{LR}$) by the SFENet ($3 \times 3$ convolution layers). The $F_0$ is then fed to the first MSB of each branch. Formally, the $F_0$ is expressed as

$$F_0 = H_3(I_{LR}) \tag{3.4}$$

where $H_3(\cdot)$ denotes the $3 \times 3$ convolution operation.

The MBNet is constructed to have $B$ branches, where each branch is a cascade of MSBs followed by their outputs' concatenation and $1 \times 1$ convolution to make a feature map. The searched MSBs in each branch have different receptive fields, and thus each branch learns multi-scale characteristics for image super-resolution. I multiply an independent scalar weight to the outputs of each node and block to adjust the gradient magnitude in back-propagation. A similar technique was used in [18]. I name these weights as gradient flow control weights and denote them as $\alpha$, as illustrated in the last part of the MBNet block in Fig. 3.1.

Formally, the output of the $d$-th MSB in the $b$-th branch, $F_{b,d}$ is

$$\begin{aligned} F_{b,d} = (\alpha_{skip})_{b,d} \times F_{b,d-1} + \\ (\alpha_{res})_{b,d} \times H_1(concat(F_{b,d,1}, \ldots, F_{b,d,M})), \end{aligned} \tag{3.5}$$

where $F_{b,d,m}$ denotes the output of the $m$-th PSN of the $d$-th multi-scale block (MSB) in the $b$-th branch, and $H_1(\cdot)$ denotes the $1 \times 1$ convolution operation for the local feature fusion layer. Also, $\alpha_{skip}$ and $\alpha_{res}$ are the gradient flow control weights for residual feature and skip connection, respectively. $F_{b,d,m}$ will be detailed in the following subsection, with Fig. 3.2 and Eq. 3.9.

Then, the output of the MBNet is a weighted sum of all the branch outputs:

$$F_{MB} = \sum_{b=1}^{B} (\alpha_{gff})_b \times (F_{gff})_b \tag{3.6}$$

where

$$(F_{gff})_b = H_1(concat(F_{b,1}, \ldots, F_{b,D})), \tag{3.7}$$

and $(\alpha_{gff})_b$ is a gradient flow control weights for global feature fusion layer. Also, $(F_{gff})_b$ is the output of global feature fusion layer of the $b$-th branch.

Finally, I obtain the reconstructed high-resolution image $I_{HR}$ by combining the up-sampled low-resolution image $I_{LR}$ and residual information in the UPNet $F_{MB}$. Formally, the $I_{HR}$ is computed as

$$I_{HR} = H_{ps}(I_{LR}) + H_{ps}(F_{MB}), \tag{3.8}$$

where $H_{ps}(\cdot)$ denotes $3 \times 3$ convolution and periodic shuffling layer as in ESPCN [14]. I fix the structure of SFENet and UPNet while searching the connection of MBNet.

### 3.3.4 Multi-scale block with partially shared Nodes

I apply a cell structure for the MSB, which means that all MSBs in the same branch have the same connection and operation. Each MSB consists of $M$ PSNs as shown in the upper part of Fig. 3.2 (a) and (b). The dashed arrows and colored arrows in Fig. 3.2 mean that these connections are to be searched. The candidate operations of the PSN are

1. $3 \times 3$ convolution,
2. $3 \times 3$ dilated convolution with rate two,
3. $3 \times 3$ dilated convolution with rate three.

Following the signal flow in Fig. 3.2, $F_{b,d,m}$ in Eq. 3.5 is calculated as

$$\begin{aligned} F_{b,d,m} = (\alpha_{skip})_{b,d,m} \times F_{b,d,m-1} + \\ (\alpha_{res})_{b,d,m} \times (H_b)_{PSN,m}(F_{b,d,m-1}), \end{aligned} \tag{3.9}$$

where $(H_b)_{PSN,m}(\cdot)$ denotes the operation of the $m$-th PSN in the $b$-th branch. The $(H_b)_{PSN,m}(\cdot)$ can be expressed as

$$(H_b)_{PSN}(\cdot) = CA(H_{(s_b)_{m,2}}(Relu(H_{(s_b)_{m,1}}(\cdot)))), \tag{3.10}$$

where $H_{(s_b)_{m,n}}(\cdot)$ denotes the $k$-th operation among $K$ candidates, which is chosen by the configuration sequence $(s_b)_{m,n}$. I construct the PSN with two operations and one Relu activation as shown in Eq. 3.10. $CA(\cdot)$ denotes channel attention layer of RCAN [34].

To reduce the number of network parameters and spread the information through the branches, the parameters of PSNs have common weights if the configuration sequence of different branches activates an identical position in their sequence. For example, if two branches' configuration sequences are '001' and '011,' the operation

corresponding to the first and the third digit share their weights. In Fig. 3.2, I emphasize the shared positions in the controller sequence (FC outputs) by big bold digits.

### 3.3.5 MBNAS

Like conventional RL-based NAS methods [19, 21], my algorithm has $\boldsymbol{\theta}$ and $\mathbf{w}$, which represents the parameter of the controller and the child network, respectively. In the search phase, $\boldsymbol{\theta}$ and $\mathbf{w}$ are trained alternately for each epoch. After the search phase is finished, I sample the sequences by the trained controller. Then, the best sequence among the sampled ones is chosen and trained from scratch.

#### Training the child network

I first train the parameters of a child network to calculate the reward signal of the controller. The problem is formulated as

$$\min_{\mathbf{w}} \mathbb{E}_{c\sim\pi(c;\boldsymbol{\theta})}[L(c;\mathbf{w})], \tag{3.11}$$

where $L(\cdot)$ denotes the loss function for the task which is the $L1$ loss in my setting. The controller's policy $\pi(c;\boldsymbol{\theta})$ is fixed when training the child network. The Adam optimizer [35] is used to optimize $\mathbf{w}$. I estimate the gradient of $\mathbb{E}_{c\sim\pi(c;\boldsymbol{\theta})}[L(c;\mathbf{w})]$ with the Monte Carlo estimate

$$\bigtriangledown_{\mathbf{w}}\mathbb{E}_{c\sim\pi(c;\boldsymbol{\theta})}[L(c;\mathbf{w})] \approx \frac{1}{M}\sum_{i=1}^{M}\bigtriangledown_{\mathbf{w}}L(c_i;\mathbf{w}), \tag{3.12}$$

where $c_i$ denotes a sampled child network by the controller's policy. I choose $M = 1$, which means that I sample just one child network for each mini batch.

#### Training the controller

In the controller training phase, $\mathbf{w}$ is fixed, and $\boldsymbol{\theta}$ is trained by REINFORCE [28] algorithm. I optimize $\boldsymbol{\theta}$ to maximize the expectation of reward signal, which can be expressed as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{P(a_{1:T};\boldsymbol{\theta})}[R], \tag{3.13}$$

where $a_{1:T}$ is the configuration sequence for the child network $c$. In the REINFORCE, the gradient of the expected reward is approximated as

$$\bigtriangledown_{\boldsymbol{\theta}} \mathbb{E}_{P(a_{1:T};\boldsymbol{\theta})}[R] = \sum_{t=1}^{T} [\bigtriangledown_{\boldsymbol{\theta}} \log P(a_t | a_{t-1:1}; \boldsymbol{\theta})(R - b)] \tag{3.14}$$

where $b$ is the baseline which is used to reduce the variance. The moving average of the reward signal is used for the baseline in my algorithm. As explained with Eq. 3.2, I use the PSNR of validation set and complexity-based penalty to calculate reward signal. Adam [35] is used to optimize the reward.

## 3.4 datasets and experiments

### 3.4.1 Settings

#### Datasets, degradation methods, and metrics

I choose DIV2K [36] dataset for the training and validation. The DIV2K dataset is widely used as a training set of various image restoration tasks. It contains 1,000 images, consisted of 800 for training, 100 for validation, and the other 100 images for test. The validation images are used as the data for measuring reward signal of controller network.

I measure the performance on four different benchmark dataset; Set5 [37], Set14 [38], BSDS100 [39], and Urban100 [40]. To compare the performances with others, I measure the PSNR and SSIM [41] of the test image on the Y channel of YCbCr color

domain. I create the synthetic low-resolution image by applying Matlab's imresize function [60].

**Implemenatation details**

I construct the controller by a two-stacked LSTM network with 64 hidden states. I connect three fully connected layers to the end of each LSTM block to get the configure sequence for the child network. I use word embedding [42] to make the input of the LSTM layer from the previous LSTM block's output.

I construct my MBNASNet with three branches ($B = 3$), four multi-branch blocks ($D = 4$), and two PSNs ($M = 2$) which have three operations as the candidate operations. The number of output feature maps for SFENet and MSNet is unified to 32. The number of intermediate features in PSNs is 128, which is four times bigger than the number of the output feature maps.

**Hyper-parameter settings**

In the search phase, I alternatively train the controller and child network for one epoch each. I initialize both the controller parameter $\boldsymbol{\theta}$ and the child network parameter $\mathbf{w}$ by using the variance scaled initialization [43] with 0.02 scaling value. I train the controller and the child network for 500 epochs. For one epoch, I apply 100 iterations for the controller, and 1,000 iterations for the child network. The learning rate of the controller is fixed to $3 \times 10^{-4}$. The learning rate of the child network initialized to $3 \times 10^{-4}$ and decreased by half for every 100 epochs. I use 16 low-resolution image patches of size $64 \times 64$ from DIV2K train images as a mini-batch of the child network. I augment the patches by randomly applying horizontal flip and 90°, 180°, 270° rotation. The $\lambda$ in Eq. 3.2 is set to 2, and $p(c; \mathbf{w})$ is the validation PSNR of child network. I randomly extract 1,000 low-resolution image patches from DIV2K validation images and compute PSNR to calculate the reward.

In the training phase, I sample 500 configuration sequences from the trained con-

Table 3.1: Mean and variance of searched networks from three controllers which are trained from different random seeds.

| Experiment | mean PSNR | mean CBP | var. PSNR |
|---|---|---|---|
| 1 | 34.134 | 0.561 | 0.00143 |
| 2 | 34.131 | 0.543 | 0.00096 |
| 3 | 34.130 | 0.554 | 0.00097 |

troller network and choose the architecture which has the best performance in the DIV2K validation set as my MBNASNet. I train the selected network for 1,000 epochs and finetune the trained network for 1,000 more epochs. The hyper-parameter settings are the same as the search phase except for the learning rate. The learning rate of the child network is initialized to $3 \times 10^{-4}$ and decreased half by 200 epochs.

### 3.4.2 Experiments on single image super-resolution (SISR)

#### MBNAS search reseult

The proposed MBNASNet has four multi-scale blocks ($D = 4$) and two PSNs ($M = 2$) with three branches ($B = 3$). I sample 500 architectures and choose the best architecture from them. For $\times 2$ scale, the configuration sequence of each branch is found to be

$$\begin{aligned} \mathbf{s}_1 &= \{0, 0, 1, 2\}, \\ \mathbf{s}_2 &= \{0, 0, 1, 2\}, \\ \mathbf{s}_3 &= \{2, 0, 1, 2\}. \end{aligned} \tag{3.15}$$

I note that my searched structure has two same blocks with different channel attention and one block with a larger receptive field to capture multi-scale features efficiently.

On the other hand, the searched configuration sequence for $\times 3$ scale is

$$\begin{aligned} \mathbf{s}_1 &= \{1, 1, 0, 2\}, \\ \mathbf{s}_2 &= \{1, 1, 0, 2\}, \\ \mathbf{s}_3 &= \{1, 1, 2, 2\}. \end{aligned} \tag{3.16}$$

The $\times 3$ scale SR task generally needs a larger receptive field than the $\times 2$ to extract multi-scale features, and my searched $\times 3$ network satisfies this property. It takes about 24 hours to train the controller and the child network by one Tesla V100 GPU in the search phase, which is far less than other NAS-based methods such as MoreMNAS [6] and FALSR [7]. To show the robustness of my search algorithm, I search three times from different random seeds. Table 3.1 indicates the mean and variance of 500 searched networks from three different controllers for $\times 3$ image super-resolution of Set5.

**Image super-resolution results**

Bicubic image down-sampling is widely used as the image degradation setting of super-resolution task. I measure PSNR and SSIM on four public benchmark dataset to compare my method with eleven state-of-the-art methods: SRCNN [1], VDSR [2], LapSRN [3], MemNet [4], MSAN [31], SelNet [33], CARN [5], $A^2F$ [61], MoreMNAS [6], FALSR [7], HNAS [45], and DeCoNASNet [47]. Among these, MoreMNAS, FALSR, DeCoNASNet, HNAS, and mine is NAS-based aproaches. HNAS uses large training patch ($96\times96$) when training and applies self-ensemble to get better performances.

The proposed NAS-based approach has efficient search algorithm, which is about twenty times faster than MoreMNAS and FALSR. For this reason, I can conduct experiments on $\times 3$ super-resolution task while other NAS-based methods do not. As shown in Table 3.2, MBNASNet performs comparable to hand-crafted state-of-the-art methods and outperforms the NAS-based methods for $\times 2$ and $\times 3$ super-resolution task.

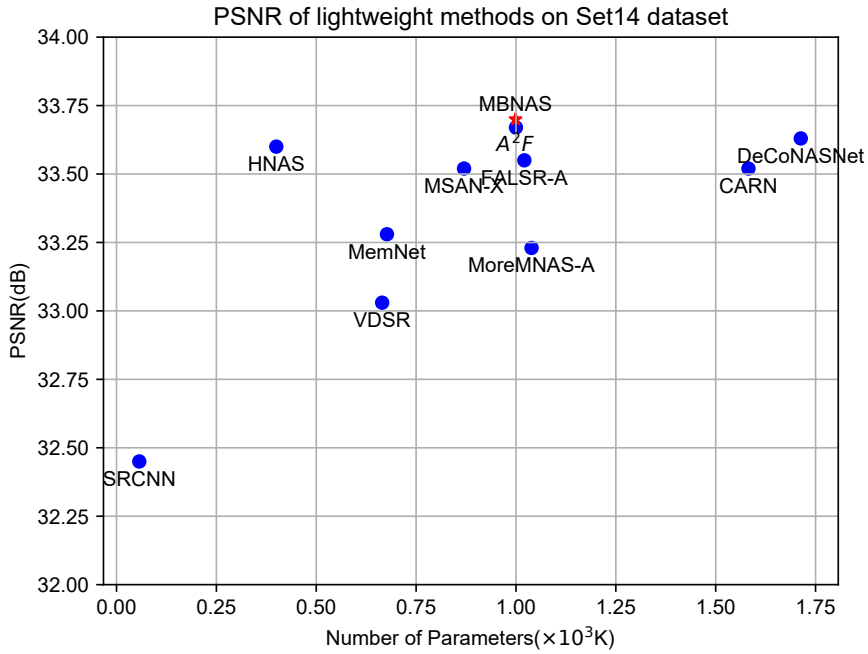



Figure 3.4: The graphical result of conventional lightweight methods and my MBNASNet on Set14 dataset. The Blue dots are conventional lightweight methods, and the red star is my MBNASNet method.

HNAS shows good performance for Set5 dataset, but MBNASNet performs better for complex datasets such as Urban100 and B100 dataset because I extract multi-scale features successfully.

Since different initial conditions may lead to different results, I perform the design four times with different initial hyperparameters. But, there are just slight differences for all the cases in Table 3.2, with PSNR variance under $10^{-4}$, validating the robustness of my method against different initial conditions. Hence, I denote the best PSNR among the four experiments, following the convention.

In Fig. 3.5 and Fig. 3.8, I display the qualitative result of my method and conventional methods. As shown in the figures, MBNASNet successfully restores the structures of the images. Specifically, my network recovers the gray vertical lines and holes in each image while other methods do not. In summary, I compare the overall $\times 2$

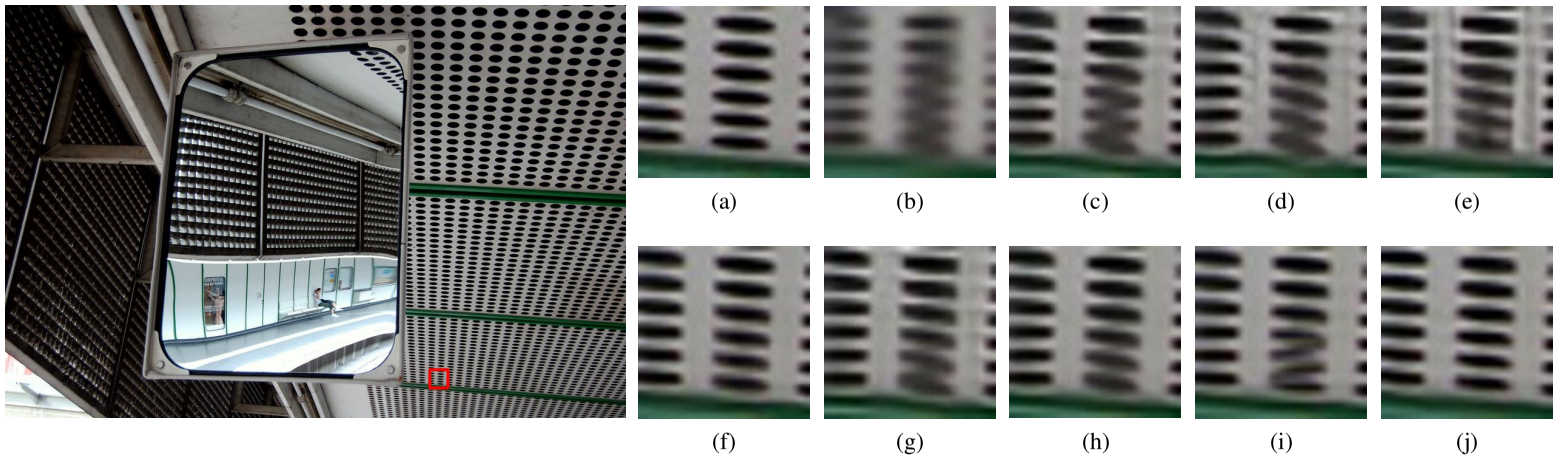



Figure 3.5: Qualitative result on the 4th image from the Urban100 dataset for $\times 2$ SR task. I compare my method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

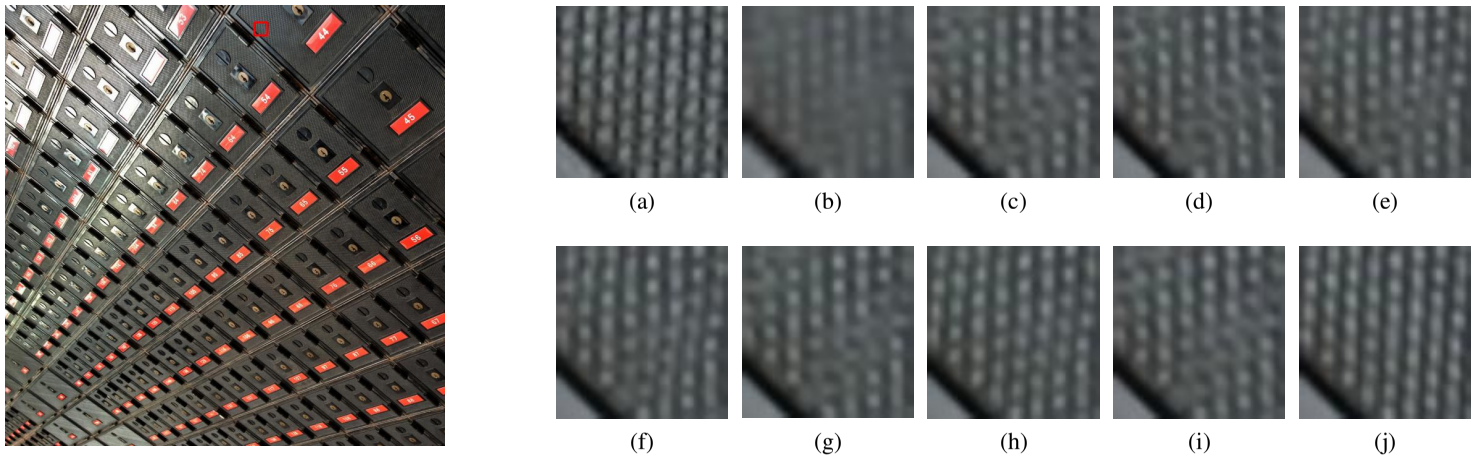



Figure 3.6: Qualitative result on the 6th image from the Urban100 dataset for $\times 2$ SR task. I compare my method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

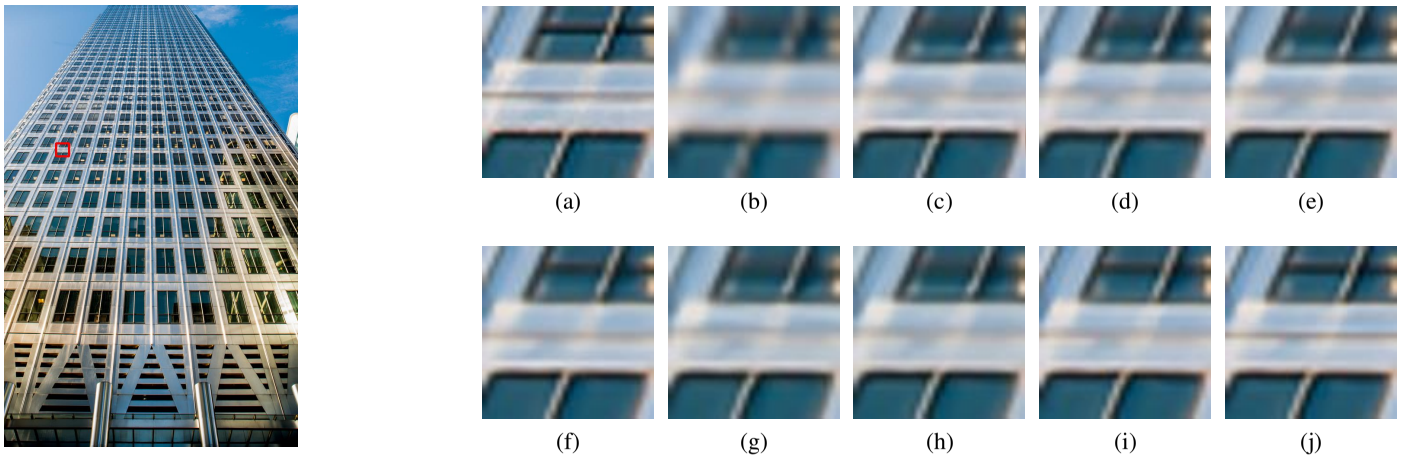



Figure 3.7: Qualitative result on the 30th image from the Urban100 dataset for $\times 2$ SR task. I compare my method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

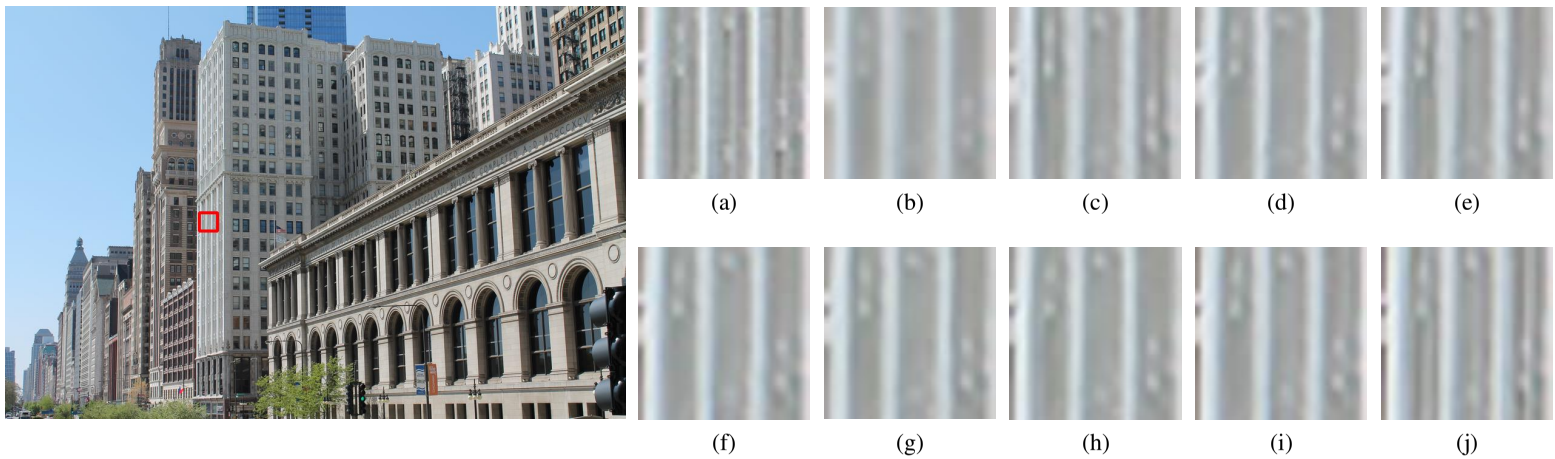



Figure 3.8: Qualitative result on the 97th image from the Urban100 dataset for $\times 2$ SR task. I compare my method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

Table 3.2: PSNR and SSIM on the public benchmark test data for $\times 2$ and $\times 3$ SR tasks. I emphasize the best and the second-best performances with the red and blue colors, respectively. Methods with bold characters are NAS-based methods, and the "Design time" at the last column indicates the times taken for the search process. All four indicated design times are calculated with the same GPU (NVIDIA Tesla V100). Other NAS-based methods do not report more than $\times 3$ SR results due to huge search times, whereas I could. *In the case of the HNAS, the complexity is an estimated one because they do not explicitly reveal the number of parameters. Also, the + sign at the HNAS denotes that they used self-ensemble, which generally gives higher PSNR than the baseline.

| Model | scale | Params | Set 5 | Set 14 | B100 | Urban100 | Design time |
|---|---|---|---|---|---|---|---|
| Bicubic | | - | 33.66 / 0.9299 | 30.24 / 0.8688 | 29.56 / 0.8431 | 26.88 / 0.8403 | - |
| SRCNN [1] | | 57K | 36.66 / 0.9542 | 32.45 / 0.9067 | 31.36 / 0.8879 | 29.50 / 0.8946 | - |
| VDSR [2] | | 665K | 37.53 / 0.9587 | 33.03 / 0.9124 | 31.90 / 0.8960 | 30.76 / 0.9140 | - |
| LapSRN [3] | | 813K | 37.52 / 0.9591 | 33.08 / 0.9130 | 31.80 / 0.8950 | 30.41 / 0.9101 | - |
| MemNet [4] | | 677K | 37.78 / 0.9597 | 33.28 / 0.9142 | 32.08 / 0.8978 | 31.31 / 0.9195 | - |
| MSAN-X [31] | | 870K | 37.86 / 0.8909 | 33.52 / 0.9167 | 32.12 / 0.8983 | 31.91 / 0.9255 | - |
| SelNet [33] | | 970K | 37.89 / 0.9598 | 33.61 / 0.9160 | 32.08 / 0.8984 | - / - | - |
| CARN [5] | $\times 2$ | 1,582K | 37.76 / 0.9590 | 33.52 / 0.9166 | 32.09 / 0.8978 | 31.92 / 0.9256 | - |
| $A^2$F-M [61] | | 1,000K | 38.04 / 0.9607 | 33.67 / 0.9184 | 32.18 / 0.8996 | 32.27 / 0.9294 | - |
| ***HNAS-C+** [45] | | ∼400K | 38.11 / 0.964 | 33.60 / 0.920 | 32.17 / 0.902 | 31.93 / 0.928 | - |
| **MoreMNAS-A** [6] | | 1,039K | 37.63 / 0.9584 | 33.23 / 0.9138 | 31.95 / 0.8961 | 31.24 / 0.9187 | 56 days |
| **FALSR-A** [7] | | 1,021K | 37.82 / 0.9595 | 33.55 / 0.9168 | 32.12 / 0.8987 | 31.93 / 0.9256 | 24 days |
| **DeCoNASNet** [47] | | 1,713K | 37.96 / 0.9594 | 33.63 / 0.9175 | 32.15 / 0.8986 | 32.03 / 0.9265 | 12 hours |
| **MBNASNet**(mine) | | 999K | 38.04 / 0.9595 | 33.70 / 0.9178 | 32.19 / 0.8992 | 32.17 / 0.9281 | 24 hours |
| Bicubic | | - | 30.39 / 0.8682 | 27.55 / 0.7742 | 27.21 / 0.7385 | 24.46 / 0.7349 | - |
| SRCNN [1] | | 57K | 32.75 / 0.9090 | 29.30 / 0.8215 | 28.41 / 0.7863 | 26.24 / 0.7989 | - |
| VDSR [2] | | 665K | 33.66 / 0.9213 | 29.77 / 0.8314 | 28.82 / 0.7976 | 27.14 / 0.8279 | - |
| MemNet [4] | | 677K | 34.09 / 0.9248 | 30.00 / 0.8350 | 28.96 / 0.8001 | 27.56 / 0.8376 | - |
| MSAN-X [31] | $\times 3$ | 1,054K | 34.19 / 0.9246 | 30.27 / 0.8403 | 29.03 / 0.8030 | 27.96 / 0.8473 | - |
| SelNet [33] | | 1,159K | 34.27 / 0.9257 | 30.30 / 0.8399 | 28.97 / 0.8025 | - / - | - |
| CARN [5] | | 1,582K | 34.29 / 0.9255 | 30.29 / 0.8407 | 29.06 / 0.8034 | 28.06 / 0.8493 | - |
| $A^2$F-M [61] | | 1,000K | 34.50 / 0.9278 | 30.39 / 0.8427 | 29.11 / 0.8054 | 28.28 / 0.8546 | - |
| **MBNASNet**(mine) | | 1,003K | 34.30 / 0.9255 | 30.25 / 0.8415 | 29.08 / 0.8042 | 28.08 / 0.8501 | 30 hours |

performance of lightweight models graphically in Fig. 3.4.

## 3.5 Discussion

In this section, I discuss the effect of the proposed method's contributions; complexity-based penalty, multi-branch structure, and partially shared parameters.

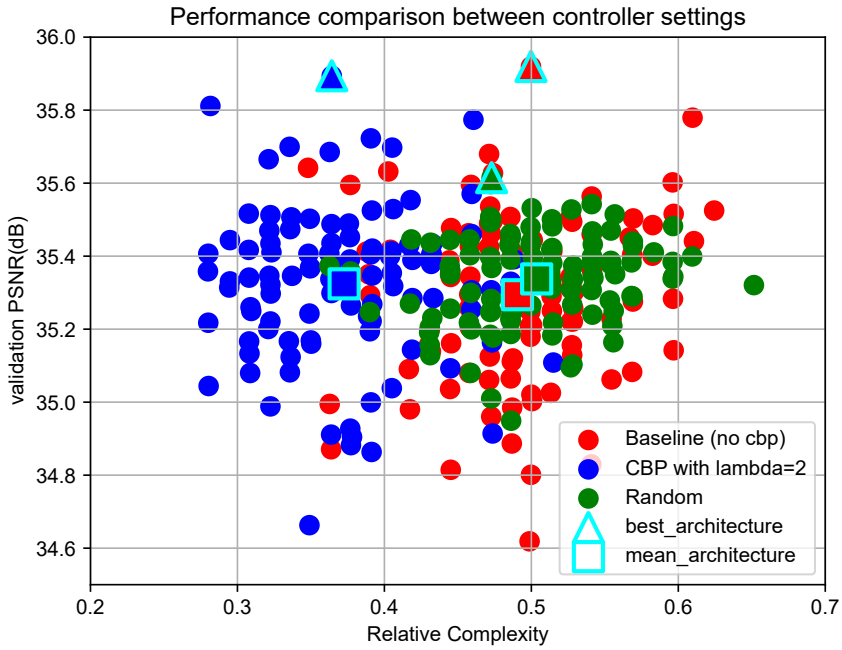



Figure 3.9: The result of three experiments for the controller. The blue dots are from the CBP, the reds are the Baseline, and the greens are Random settings. The ”Relative Complexity” is defined the same as cbp in equation(3.3), meaning the cbp in the case of NAS design results. In the case of random and baseline, since the ”penalty” is not defined, I denote it as ”Relative Complexity.”

### 3.5.1 Effect of the complexity-based penalty to the performance of controller

To evaluate the controller's performance and the effect of complexity-based penalty in the search phase, I conduct three experiments.The first experiment uses a non-trained controller, which generates a random controller sequence (denoted as Random). The controller trained with the PSNR reward but without the complexity-based penalty is denoted as Baseline, and the one including the complexity-based penalty is denoted as CBP. I choose $\lambda = 2$ for the complexity-based penalty.

I sample 100 structures for each controller setting and measure the average and the best performance, as shown in Table 3.3. Also, their distributions are illustrated in

Table 3.3: Performance comparison between controller settings.

| Search setting | | Random | Baseline | CBP |
|---|---|---|---|---|
| Best | PSNR | 35.62 | **35.92** | 35.89 |
| | Penalty | 0.473 | 0.500 | **0.364** |
| Mean | PSNR | **35.34** | 35.29 | 35.32 |
| | Penalty | 0.504 | 0.491 | **0.373** |

Table 3.4: PSNR of MBNASNet with/without gradient flow control weights $\alpha$ on the public benchmark test data for $\times 2$ SR tasks. I emphasize the difference between two experiment by blue texts. I train each architecture for 1000epochs.

| Model | scale | Set 5 | Set 14 | B100 | Urban100 |
|---|---|---|---|---|---|
| MBNASNet without $\alpha$ | $\times 2$ | 37.9426(-0.034) | 33.5727(-0.050) | 32.1357(-0.006) | 31.9743(-0.052) |
| MBNASNet with $\alpha$ | | 37.9767 | 33.6230 | 32.1413 | 32.0264 |

Fig. 3.9, where blue dots are the results of the CBP with $\lambda = 2$, red dots correspond to the Baseline, and the greens to the Random. I can see that the Baseline setting finds better architectures than the Random in terms of PSNR, sometimes with increased complexity. On the other hand, the CBP setting successfully generates lightweight sequences that have comparable PSNR to the Baseline.

### 3.5.2 Effect of multi-branch structure and partial parameter sharing scheme

To compare and visualize the effect of multi-branch structure and partial parameter sharing (PPS) scheme, I create three networks; single-branch, multi-branch without PPS, multi-branch with PPS. I set the parameters of three experiments by $\sim 1,000K$ to fairly compare the results.

I measure the PSNR of each structure on the Set5 dataset. Fig. 3.10 shows the results of three structures for 400 epochs. I can find that the multi-branch structure converges faster than the single branch structure. Furthermore, with the partial parameter sharing scheme, I can successfully overcome the performance degradation

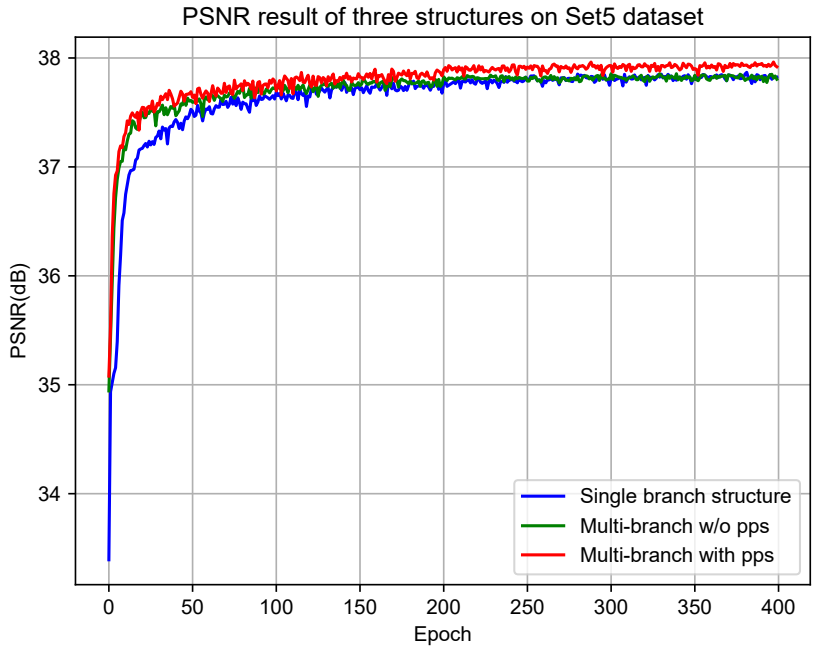



Figure 3.10: The PSNR on Set5 for three structures. The red line indicates my MBNASNet structure, the green line is the multi-branch structure with separate parameters, and the blue is the single branch structure.

phenomenon in the multi-branch structure.

### 3.5.3 Effect of gradient flow control weights and complexity-based penalty coefficient

Gradient flow control weights allow MBNASNet to overcome the gradient vanishing problem by adjusting the gradient magnitude in the back-propagation process. I train MBNASNet with/without gradient flow control weights $\alpha$ and compare their performance in Table 3.4 and Fig. 3.11. The results show that $\alpha$ helps the MBNASNet converge to better point and achieve better performance.

To compare the effect of CBP weight $\lambda$, I train the controller with different $\lambda$ values ($\lambda = 0.5, 1, 2, 4$) and compare their search results in Table 3.5. I can see that the mean

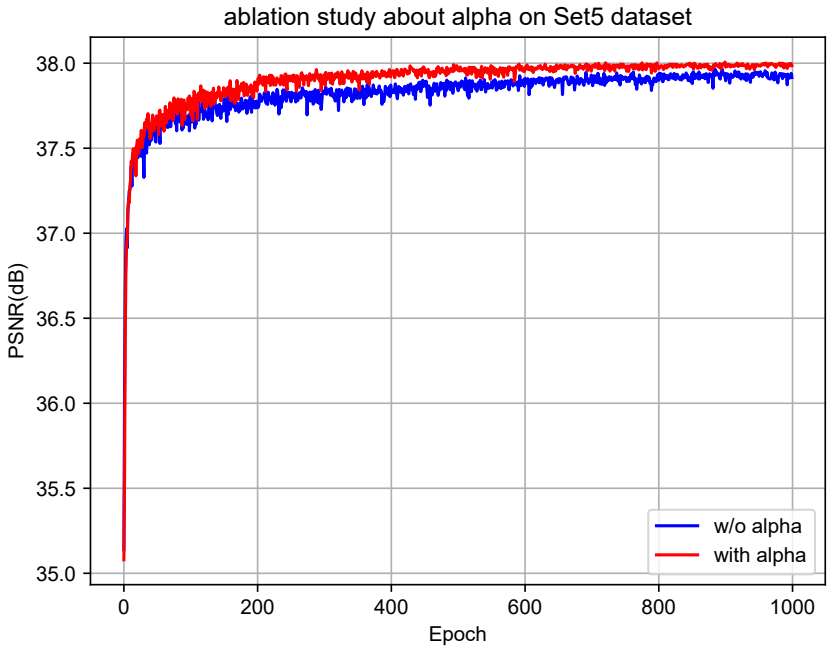



Figure 3.11: The PSNR on Set5 of MBNASNet architecture with/without gradient flow control weights $\alpha$. The red line indicates my MBNASNet structure with $\alpha$, and the blue is MBNASNet without $\alpha$.

CBP value tends to decrease (a lighter network is found), and the mean PSNR slightly decreases as the $\lambda$ becomes larger. When the $\lambda$ becomes too big ($\lambda = 4$), the controller fails to find a promising network in the search space. The experiments validate that the $\lambda$ efficiently controls the trade-off between the performance and the number of parameters until $\lambda = 2$, and hence I use $\lambda = 2$ in other experiments.

## 3.6 Summary

I have proposed a new NAS-based SR network, named as MBNASNet. I have attempted to improve the performance of the NAS-based SR by adopting a multi-branch network that can extract multi-scale features. In other words, I could obtain a better SR model by expanding the search space. I also regularized the reward signal of RE-

Table 3.5: Mean PSNR and complexity-based penalty on different $\lambda$. The PSNR is calculated by Set5 benchmark dataset.

| Experiment | mean PSNR | mean CBP |
|---|---|---|
| $\lambda = 0.5$ | 37.830 | 0.569 |
| $\lambda = 1$ | 37.829 | 0.561 |
| $\lambda = 2$ | 37.829 | 0.557 |
| $\lambda = 4$ | 37.801 | 0.575 |

INFORCE algorithm with a complexity-based penalty to favor a lightweight network. Besides, the partial parameter sharing scheme successfully reduces the number of parameters and helps the information transfer between each branch. It takes 24 hours to find promising network structures, which is a lot faster than the existing NAS-based design methods. The results show that the proposed method performs comparably to the conventional hand-crafted structures and other NAS-based networks. I will release my codes and more result images at https://github.com/Junem360/MBNASNet.

# Chapter 4

# Meta-transfer learning for simultaneous search of various scale image super-resolution

## 4.1 Introduction

restoration is widely used as a preprocessing algorithm of various vision tasks. Specifically, single image super-resolution (SISR) which recovers high resolution (HR) results from low resolution (LR) images is applied to the tasks such as medical image analysis [8], security image processing [10], satellite image recognition [9]. Recently, researchers have been proposed deep-learning based methods [1–5, 13–18] and they performs better than the conventional SISR methods [11, 12].

In general, the performance of deep-learning based SR methods increases as the network gets deeper or more complex. However, it needs tremendous amount of atempts and adjustments to find optimal structures when the structure goes deeper and complicated. To reduce the efforts involved in these trial and error, the neural architecture search (NAS) algorithm has been developed [19–26, 44].

Earlier NAS methods [19, 20, 23] concentrate on the image classification task and they automate the network construction and evaluation process to decrease the time used to create the optimal network for the purpose. Because It takes a long time to

search optimal architectures by these methods, some researchers have developed efficient neural architecture search algorithms to reduce the search time. Specifically, ENAS [21] and DARTS [25] adopt directed acyclic graph (DAG) and weight sharing scheme and dramatically reduce the time for network search process.

Neural architecture search methods have been expanded to the image super-resolution task [6,7,45,47] (MoreMNAS, FALSR, HNAS, DeCoNAS). MoreMNAS and FALSR use reinforced evolution algorithm but they suffer from their long search time. To reduce the search time, HNAS, DeCoNAs and MBNAS applies efficient neural architecture search [21] and find optimal networks in a short time. However, existing NAS-SR method regard different scale super-resolution as an independent tasks, and network search for arbitrary scale image super-resolution is not supported. Furthermore, when I use weight sharing scheme to find optimal structure, there is a difference in performance between evaluation and search phase. This phenomenon, so called search-evaluation gap, causes the final performance degradation of the structures selected by weight sharing architecture search method.

To simultaneously search the networks for various scale image super-resolution task and alleviate the search-evaluation gap, I apply meta-transfer learning to the neural architecture search process. Specifically, I adopt model-agnostic meta-learning [62] when searching for SR network. The meta-learning algorithm trains the controller network to find optimal state that adapt fast to various scale image super-resolution task. Also, the parameter of child network is trained by meta-learning scheme and the performance of the child network in the search process approaches the actual performance in evaluation phase.

My main contributions are summarized as follows:

1. New NAS-based SR: I propose a meta-learning based SR network design method, named MetaNAS, which searches for the networks of various scale image super-resolution task simultaneously.

2. Reduced search-evaluation gap : I train the child network with model-agnostic

meta-learning algorithm to apply the actual performance in search phase and reduce the search-evaluation gap phenomenon.

The rest of this chapter is organized as follows. I summarize the related works about SISR, NAS and meta-learning methods in section 2. In section 3, I explain my proposed MetaNAS for SISR. I precisely explain the details about my algorithm and the implementation, followed by experiment results. Finally, I provide a summary and conclusion in section 5.

## 4.2 Related Work

### 4.2.1 Single image super-resolution

Researchers have been proposed a number of deep learning based SR methods [1–5, 13–18]. As a pioneer work, Dong *et al.* [1] (SRCNN) proposed the network consist of three shallow convolutional layers and outperformed the conventional SR methods. FRCNN [13] and ESPCN [14] tried to reduce the feature map size and channel number by deconvolution and pixel-shuffle layer. VDSR [2] used very deep convolutional neural network by adopting residual learning and gradient clipping strategy. Lim *et al.* [15] increased the PSNR by proposing an internal residual learning with extensive features (EDSR) and multi-scale structure (MDSR). MemNet [4], MSRN [18], and DenseSR [16] proposed specific block architectures to achieve better performance. SelNet [33] proposed selection unit and replacing the non-linear operation such as Relu. Zhang *et al.* proposed feature fusion layer and residual dense block in their paper [17] (RDN). RCAN [34] proposed a channel attention scheme and select important feature channels among the output features, which successfully increase the PSNR result of the SR network.

### 4.2.2 Neural architecture search

There are a number of hyper-parameters to create a deep neural architecture. For example, I need the number of layers, channels, operations, connections, etc. to create the deep learning architecture. Researchers have designed a lot of deep neural networks and evaluate them to find optimal architecture. Because this manual network design and evaluation is a laborious task, researchers have been proposed the methods to automate these processes, so called neural architecture search (NAS) [19–26].

At first, most of neural architecture search methods focus on simple image classification task. Zoph *et al.* [19] proposed a reinforcement learning (RL) based NAS algorithm. At first, they designed A controller network and a child network. controller network was consist of LSTM and generated the configure sequence for the child network. the child network was created by the configure sequence from controller network and evaluated. The controller network was trained by REINFORCE [28], which is a kind of policy gradient algorithm. The performance of the child network was used as a reward signal of the REINFORCE algorithm. Because they trained the child network from scratch, it took a huge amount of time to evaluate performance and calculate the reward. To reduce the time to get a reward signal, PNAS by Liu *et al.* [20] applied the sequential model-based optimization (SMBO) and proposed a surrogate model which predicts the network's performance. On the one hand, Pham *et al.* [21] proposed efficient NAS algorithm that shares the weight of child networks during search phase to reduce the evaluation time. This method regard the search space as a super-graph and configured each child network as a sub-graph of the search space. The trained parameters of the child network were shared during the search phase and stored their weights in the super-graph.

Evolutionary methods [22–24] are another main stream of the NAS. They selected a hundred architectures randomly and encoded each network to binary codes. After that, crossover or mutations are applied to the sequence and evaluate them to find a promising structure. Lu *et al.* [23] applied Bayesian optimization algorithm to the

evaluation process to take an advantage from previous search result. AmoebaNet [24] applied an aging evolution algorithm to NAS to discard the old network.

There were some methods that adopted different algorithms from RL and evolutionary methods [25, 26, 44, 48] (DARTS, SGAS, NAO, CSA-NAS). Specifically, DARTS proposed gradient-based neural architecture search that applied continuous relaxation to the neural architecture's connections to find optimal network and the filter weight simultaneously by gradient descent algorithm. SGAS adopted a greedy algorithm to select operation on the DARTS baseline and found the optimal network without retraining from scratch. NAO constructed a neural network which was trained to project the configure code to the embedding space and chose the best architecture as a result. CSA-NAS adopts a binary crow search algorithm to find the optimal architecture.

### 4.2.3 Image super-resolution with neural architecture search

There are some papers that expanded NAS method to image super-resolution task [6, 7, 45]. MoreMNAS [6] and FALSR [7] adopted reinforced evolution algorithm to find optimal SR network. HNAS [45] applied efficient NAS [21] with a hierarchical search algorithm that finds optimal cell structure and feature-upsampling positions. They also considered the computational cost (FLOPS) to meet the resource budget.

However, above mentioned NAS-SR methods have some drawbacks. First of all, they can find an optimal structure for the single scale image super-resolution task at a time. This disadvantages critical for real-world image super-resolution. Furthermore, fast NAS-SR algorithms such as HNAS adopt weight sharing scheme in search phase. But the result of trained weights of shared parameters is differ from the non-shared parameters. This causes the search-evaluation gap that vary in performance during search and measurement. To alleviate these problems, I combine meta-learning with NAS-SR formulation. Specifically, I adopt model-agnostic meta-learning method when training controller and child network in search phase. As a result, The controller is trained to

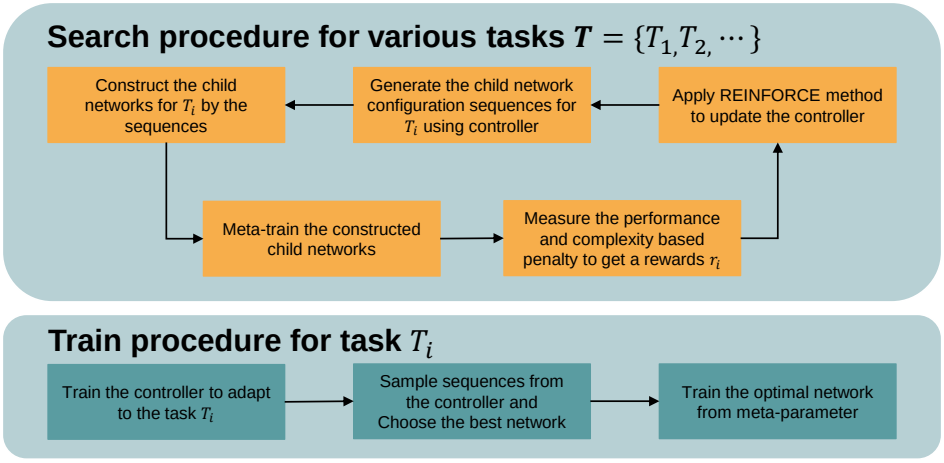



Figure 4.1: The overall procedure of MetaNAS. In the search phase, the controller and constructed child network are trained alternatively with meta-learning scheme. In the training phase, the searched final architecture is trained from pre-trained parameter.

adapt fast to various scale image super-resolution tasks. Also, I conduct several updates when calculating reward from child network to reduce the search-evaluation gap.

### 4.2.4 Meta-learning

Various meta-learning algorithms have been proposed recently. They can be classified to three types; metric-based, memory network-based and optimization-based methods. Metric-based algorithms [63–65] try to learn metric space that efficiently trained by few samples. Memory network-based methods [66–68] trains memory network to generalize well to unseen tasks. Optimization-based algorithms [62, 69–71] use gradient descent directly to the meta-learner optimization. I choose MAML [62] as my meta-learning method, which is the kind of optimization-based algorithms.

## 4.3 Method

The overall procedure of my MetaNAS is shown in Fig. 4.1. To search the optimal architecture, I design the same search space, child network and controller as MBNASNet

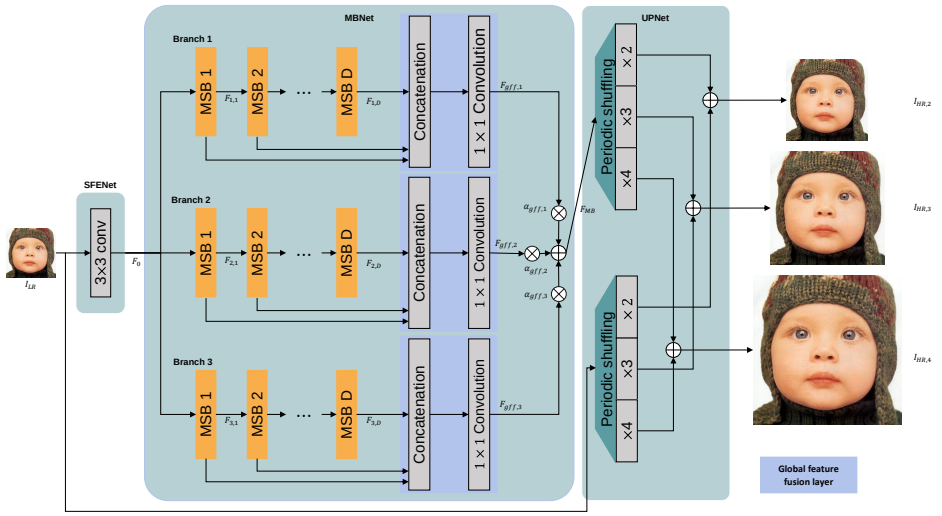



Figure 4.2: The child network structure of used in my method. It consists of a shallow feature extraction network (SFENet), a multi-branch network (MBNet), and an upscaling network (UPNet). The result of each branch is combined and upsampled by the periodic shuffling layer.

(chap. 3). Fig. 4.2 shows the child network structure of MBNAS used in my proposed method. I apply MAML [62] algorithm to simultaneously search for various scale image super-resolution task as shown in Fig. 4.3. Specifically, my method consist of three steps: meta-learning, meta-transfer learning and transfer learning.

In meta-learning step, I train the parameters of child network $\mathbf{w}$ and controller network $\boldsymbol{\theta}$ to be sensitive and transferable to various scale image super-resolution task. Next, I again train the $\boldsymbol{\theta}$ and $\mathbf{w}$ to adapt well to the specific structure of selected scale image super-resolution in meta-transfer learning step. Finally, I choose the optimal structure from trained controller and train it from pre-trained parameter $\mathbf{w}$ to evaluate the final performance.

### 4.3.1 Meta-learning

In this step, I seek to find the sensitive initial point for the parameters of the controller and child network, $\boldsymbol{\theta}_m$ and $\mathbf{w}_m$, where adapts well to the specific scale image super-

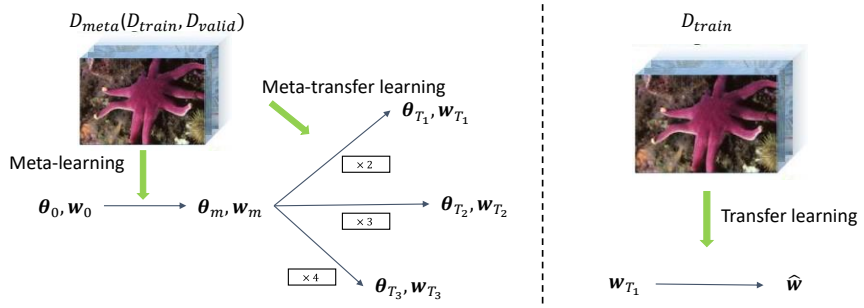



Figure 4.3: The overall meta-learning procedure of my proposed MetaNAS. From random initial point $\boldsymbol{\theta_0}$ and $\mathbf{w_0}$, meta-learning is applied to seek $\boldsymbol{\theta}_m$ and $\mathbf{w}_m$. Meta-transfer learning is conducted to get sensitive and transferable parameters with regard to the scale and the structure.

resolution with a few gradient update. Inspired by MAML [62], I follows meta-training of MAML with some differences.

I create the dataset $\mathcal{D}_{meta}$ for meta-learning step. $\mathcal{D}_{meta}$ consist of high-resolution and low-resolution image pairs of various scale. Specifically, I use three scale ($\times 2$, $\times 3$, $\times 4$) LR/HR image pairs with same probability. To calculate the reward signal and update the controller parameter $\boldsymbol{\theta}$, I should train the parameters of a child network first. Without meta-learning, the child network's parameter $\mathbf{w}$ is updated as

$$\mathbf{w_{i+1}} = \mathbf{w_i} - \alpha \bigtriangledown_{\mathbf{w}} L(c_i; \mathbf{w}) \tag{4.1}$$

where $L(\cdot)$ denotes the loss function for the task which is the $L1$ loss and $c_i$ denotes a sampled child network by the controller's policy. To apply MAML meta-training shceme to Eq. 4.1, I divide $\mathcal{D}_{meta}$ to two groups: $\mathcal{D}_{tr}$ for task-level training and $\mathcal{D}_{te}$ for task-level test. I apply several gradient descent updates to $\mathbf{w}$ to conduct adaptation to a new task $\mathcal{T}_j$ and one update process is expressed as

$$\mathbf{w}_j = \mathbf{w} - \alpha \bigtriangledown_{\mathbf{w}} L^{tr}_{\mathcal{T}_j}(c_i; \mathbf{w}), \tag{4.2}$$

where $\alpha$ is the task-level learning rate. After that, the child network's parameters $\mathbf{w}$

are trained to reduce test error of $\mathcal{D}_{te}$ with respect to $\mathbf{w_j}$. Eventually, parameter update is expressed as

$$\mathbf{w} \leftarrow \mathbf{w} - \beta \bigtriangledown_{\mathbf{w}} \sum_{\mathcal{T}_j \sim p(\mathcal{T})} L^{te}_{\mathcal{T}_j}(c_i; \mathbf{w}_j), \tag{4.3}$$

where $\beta$ is the meta-learning rate.

After updating child network parameters for one epoch, Ie train the controller parameters $\boldsymbol{\theta}$. I calculate task-specific reward $R_j$ as

$$R_j = p(c; \mathbf{w}_j) - \lambda \times cb(c), \tag{4.4}$$

where $cb(c)$ is the complexity-based penalty of MBNAS (chap. 3) and $\mathbf{w}_j$ is the child network parameter which is adapted to task $\mathcal{T}_j$.

Finally, the gradient update of the expected reward is defined as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \bigtriangledown_{\mathbf{w}} \sum_{\mathcal{T}_j \sim p(\mathcal{T})} \sum_{t=1}^{T} [\bigtriangledown_{\boldsymbol{\theta}} \log P(a_t | a_{t-1:1}; \boldsymbol{\theta})(R_j - b)], \tag{4.5}$$

where $a_{1:T}$ is the configuration sequence for the child network $c$ and $b$ is the baseline which is used to reduce the variance.

### 4.3.2 Meta-transfer learning

After finding the intermediate parameters $\mathbf{w}_m$ and $\boldsymbol{\theta}_m$, I focus on one specific scale image super-resolution task and conduct adaptation of $\mathbf{w}_m$ and $\boldsymbol{\theta}_m$ to make them sensitively transform to the selected scale. The adaptation procedure is almost same as meta-learning step, but the input and output data pair is fixed to the selected scale. Because $\mathbf{w}_m$ and $\boldsymbol{\theta}_m$ is already sensitive to the various scale image super-resolution task, I only need the small number of updates to find $\mathbf{w}_{\mathcal{T}_i}$ and $\boldsymbol{\theta}_{\mathcal{T}_i}$, which is the final parameters of specific scale image super-resolution task.

### 4.3.3 Transfer-learning

In this step, I find the optimal network for image super-resolution task by trained controller and child network. I sample the architectures from the controller and evaluate each structure to choose the best network. Different from DeCoNAS and MBNAS (chap. 3), I conduct an adaptation to the child network's parameter to alleviate the search-evaluation gap. Also, I use $\mathbf{w}_{\mathcal{T}_i}$ as the initial point of the final evaluation to get better result.

## 4.4 datasets and experiments

### 4.4.1 Settings

#### Datasets,and metrics

I use DIV2K [36] as a dataset for the training. The DIV2K dataset is widely used as a training set of various vision tasks. It consist of 1,000 images, 800 for training, 100 for validation, and the other 100 images for test. The training images are used as $\mathcal{D}_{tr}$ and the validation images are used as $\mathcal{D}_{te}$ in meta-learning process.

I measure the PSNR and SSIM [41] on four benchmarks to compare performance with other methods; Set5 [37], Set14 [38], BSDS100 [39], and Urban100 [40]. I measure the PSNR and SSIM of the test image on the Y channel of YCbCr color domain. The low-resolution image is created by Matlab's imresize function [60].

#### Implemenatation details

For the controller and child network construction, I adopt the setting of MBNAS. The controller consist of LSTM network with 64 hidden states. The child network also follows MBNASNet structure with three branches, four multi-branch blocks and two PSNs. The number of output feature channel is 32 and the number of intermediate features in PSNs is 128.

**Hyper-parameter settings**

In the meta-learning and meta-transfer learning step, the controller and child network are trained alternatively for one epoch. The initial point of two structure, $\boldsymbol{\theta}_0$ and $\mathbf{w}_0$ is initialized by variance scaled initialization [43] with 0.02 scaling value. I train the controller and the child network for 500 epochs in meta-learning step, and five epochs for meta-transfer learning step. For one epoch, I apply 1000 iterations for the controller, and 1,000 iterations for the child network. The task-level learning rate $\alpha$ is set to 0.01 and the meta-learning rate $\beta$ for controller is fixed to $3 \times 10^{-3}$. The meta-learning rate of the child network initialized to $3 \times 10^{-4}$ and decreased by half for every 100 epochs. The bath size of child network is 16 and the patch size of low-resolution image is $64 \times 64$. I use data augmentation that the patches are applied random horizontal flip and 90°, 180°, 270° rotation. The $\lambda$ in complexity based penalty is set to 2.

For the final evaluation, I sample 100 configuration sequences from the trained controller and choose the optimal architecture which has the best PSNR on the DIV2K validation set. I train the selected network for 1,000 epochs and finetune the trained network for 1,000 more epochs. The hyper-parameter settings are the same as the search phase except for the learning rate. The learning rate is initialized to $3 \times 10^{-4}$ and decreased half for every 200 epochs.

### 4.4.2 Experiments on single image super-resolution (SISR)

**Experiment about search-evaluation gap**

I conduct the experiment to verify that my proposed algorithm alleviates the search-evaluation gap compared to the existing method. I randomly sample 11 architectures from controller and train all of them from scratch to get an exact evaluation performance of each network. Then, I evaluate the sampled networks with three controllers. The first controller is trained by baseline training. Second and third controller is trained by transfer learning and meta-learning. As shown in the Fig. 4.4, the meta-learned con-

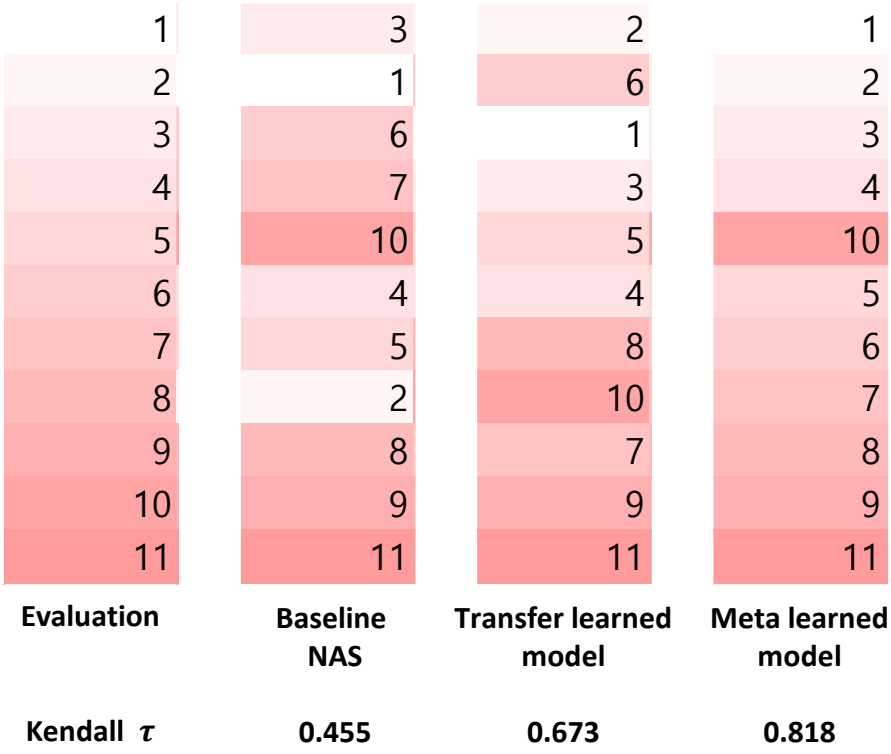



Figure 4.4: The experiment about search-evaluation gap for each controller. I calculate the kendall rank correlation coefficient to compare each settings by value.

troller's search evaluation gap is smaller than others. To compare the search-evaluation gap as a value, I calculate the kendall rank correlation coefficient. As expected, the meta-learned controller has the highest value compared to the other two controllers.

**MetaNAS search result**

It takes about two hours to meta-transfer learning for specific scale image super-resolution. I illustrate the search result of $\times 2$ and $\times 3$ image super-resolution task on Fig. 4.5. As shown in the figure, I can find that the controller successfully adapted to each image super-resolution task in a short time.

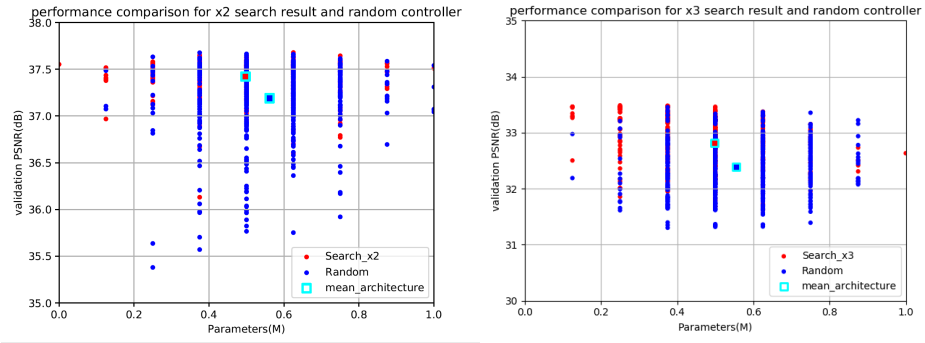



Figure 4.5: The experiment about simultaneous search for various scale image super-resolution task. By short time of adaptation process, the controller learns to sample promising networks for each scale.

### Image super-resolution results

I measure PSNR and SSIM on four public benchmark dataset to compare my method with eleven state-of-the-art methods: SRCNN [1], VDSR [2], MemNet [4], MSAN [31], SelNet [33], CARN [5], $A^2F$ [61], MoreMNAS [6], FALSR [7], HNAS [45], DeCoNASNet [47] and MBNASNet (chap. 3). Among these, MoreMNAS, FALSR, DeCoNASNet, HNAS, MBNASNet and mine is NAS-based aproaches. HNAS uses larger training patch ($96\times96$) than others and also applies self-ensemble scheme to increase the performances.

As shown in Table 4.1, my proposed network performs comparable to hand-crafted state-of-the-art methods and outperforms the NAS-based methods.

In Fig. 4.6, I compare the qualitative result of my method with existing methods. MetaNASNet successfully restores the details of the images as shown in the figure.

## 4.5 Summary

I have proposed a new NAS-based SR network with meta-learning scheme, named as MetaNASNet. I have attempted to alleviate the search-evaluation gap in the search phase by applying model-agnostic meta-learning method. I also conduct simultane-

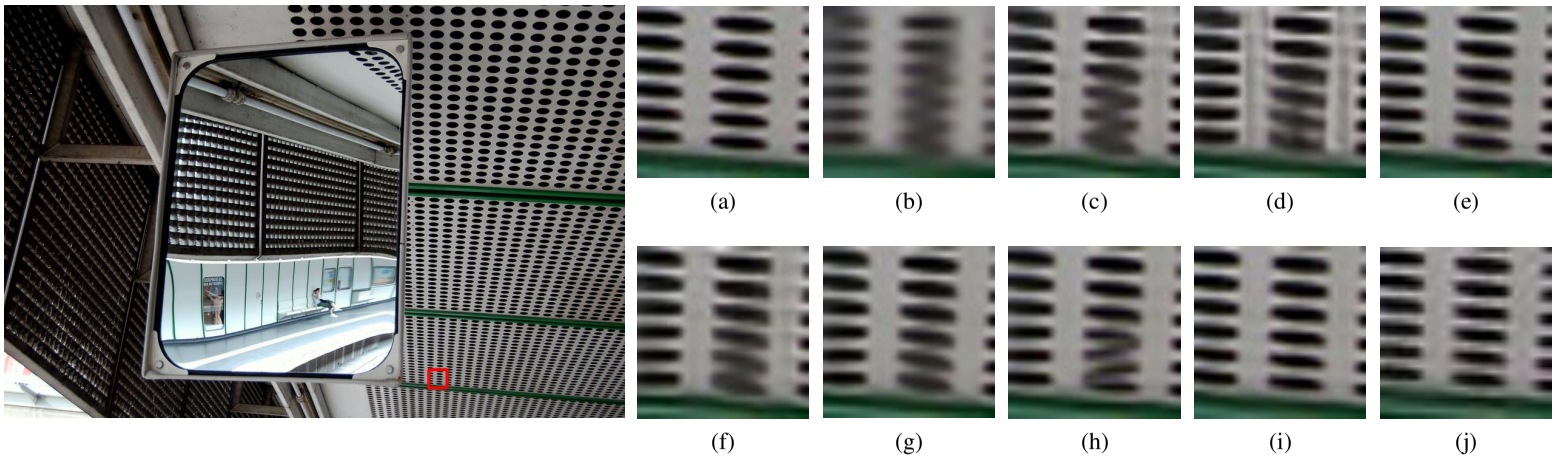



Figure 4.6: Qualitative result on the 4th image from the Urban100 dataset for $\times 2$ SR task. I compare my method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) MemNet. (e) CARN. (f) MoreMNAS. (g) FALSR. (h) DeCoNASNet. (i) MBNASNet. (j) Proposed.

Table 4.1: PSNR and SSIM on the public benchmark test data for $\times 2$ SR tasks. I emphasize the best and the second-best performances with the red and blue colors, respectively. Methods with bold characters are NAS-based methods, and the "Design time" at the last column indicates the times taken for the search process. All the indicated design times are calculated with the same GPU (NVIDIA Tesla V100). The + sign at the HNAS denotes that they used self-ensemble, which generally gives higher PSNR.

| Model | scale | Params | Set 5 | Set 14 | B100 | Urban100 | Design time |
|---|---|---|---|---|---|---|---|
| Bicubic | | - | 33.66 / 0.9299 | 30.24 / 0.8688 | 29.56 / 0.8431 | 26.88 / 0.8403 | - |
| SRCNN [1] | | 57K | 36.66 / 0.9542 | 32.45 / 0.9067 | 31.36 / 0.8879 | 29.50 / 0.8946 | - |
| VDSR [2] | | 665K | 37.53 / 0.9587 | 33.03 / 0.9124 | 31.90 / 0.8960 | 30.76 / 0.9140 | - |
| LapSRN [3] | | 813K | 37.52 / 0.9591 | 33.08 / 0.9130 | 31.80 / 0.8950 | 30.41 / 0.9101 | - |
| MemNet [4] | | 677K | 37.78 / 0.9597 | 33.28 / 0.9142 | 32.08 / 0.8978 | 31.31 / 0.9195 | - |
| MSAN-X [31] | | 870K | 37.86 / 0.8909 | 33.52 / 0.9167 | 32.12 / 0.8983 | 31.91 / 0.9255 | - |
| SelNet [33] | | 970K | 37.89 / 0.9598 | 33.61 / 0.9160 | 32.08 / 0.8984 | - / - | - |
| CARN [5] | $\times 2$ | 1,582K | 37.76 / 0.9590 | 33.52 / 0.9166 | 32.09 / 0.8978 | 31.92 / 0.9256 | - |
| $A^2$F-M [61] | | 1,000K | 38.04 / 0.9607 | 33.67 / 0.9184 | 32.18 / 0.8996 | 32.27 / 0.9294 | - |
| **HNAS-C+** [45] | | ∼400K | 38.11 / 0.964 | 33.60 / 0.920 | 32.17 / 0.902 | 31.93 / 0.928 | - |
| **MoreMNAS-A** [6] | | 1,039K | 37.63 / 0.9584 | 33.23 / 0.9138 | 31.95 / 0.8961 | 31.24 / 0.9187 | 56 days |
| **FALSR-A** [7] | | 1,021K | 37.82 / 0.9595 | 33.55 / 0.9168 | 32.12 / 0.8987 | 31.93 / 0.9256 | 24 days |
| **DeCoNASNet** [47] | | 1,713K | 37.96 / 0.9594 | 33.63 / 0.9175 | 32.15 / 0.8986 | 32.03 / 0.9265 | 12 hours |
| **MBNASNet** (chap. 3) | | 999K | 38.04 / 0.9595 | 33.70 / 0.9178 | 32.19 / 0.8992 | 32.17 / 0.9281 | 24 hours |
| **MetaNASNet** (mine) | | 1,151K | 38.05 / 0.9597 | 33.70 / 0.9181 | 32.20 / 0.8993 | 32.20 / 0.9280 | 40 hours |

ous search of various scale image super-resolution task by adopting meta-learning and adaptation. It takes 40 hours to find promising network structures for $\times 2, 3, 4$ image super-resolution task, which is a lot faster than the existing NAS-based design methods. The results show that the proposed method performs comparably to the conventional hand-crafted structures and other NAS-based networks.

# Chapter 5

# Conclusion

I have proposed a new image super-resolution (SR) method using neural architecture search (NAS) algorithm in this dissertation. In chapter 2, I have proposed a reinforcement-learning based NAS algorithm for image SR, named as DeCoNAS. I show that the proposed method find an optimal lightweight SR network with fast search time. I also proposed a feature fusion search strategy to find optimal feature fusion layer connection, which verified the importance of global/local fusion connection for the SR. Furthermore, the proposed complexity-based penalty in reward signal efficiently reduce the network complexity and enabled searching lightweight network architecture. Experiments show that the DeCoNASNet achieves higher performance with regard to the computational complexity and the PSNR when compared to the existing SR netoworks.

In chapter 3, I have proposed a new NAS-based multi-branch SR network, named as MBNASNet. I have tried to increase the PSNR/SSIM of the NAS-SR method by applying a multi-branch network that can extract multi-scale features. I expand the search space from single branch to the multi branch structure and obtain a better SR network. Moreover, I partially shared the parameters in the network structure to increase information transferablity and to reduce the number of parameters. It takes a day to find optimal network structures, which is a lot faster than the existing NAS-based design

methods. The results show that the proposed method outperforms existing NAS-based networks.

In chapter 4, I have proposed a new NAS-based SR network with meta-learning scheme, named as MetaNASNet. I adopt model-agnostic meta-learning scheme to alleviate the search-evaluation gap problem. I also achieve simultaneous search of various scale image super-resolution task by an adaptation procedure. I find optimal structures for $\times 2, 3, 4$ image super-resolution task simultaneously in 40 hours. The results show that the proposed method reduce the search-evaluation gap compared to the existing NAS-SR methods.

# Bibliography

[1] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[2] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[3] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 624–632.

[4] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4539–4547.

[5] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 252–268.

[6] X. Chu, B. Zhang, and R. Xu, "Multi-objective reinforced evolution in mobile neural architecture search," in *European Conference on Computer Vision*. Springer, 2020, pp. 99–113.

[7] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 59–64.

[8] J. S. Isaac and R. Kulkarni, "Super resolution techniques for medical image processing," in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*. IEEE, 2015, pp. 1–6.

[9] Y. Luo, L. Zhou, S. Wang, and Z. Wang, "Video satellite imagery super resolution via convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 12, pp. 2398–2402, 2017.

[10] W. W. Zou and P. C. Yuen, "Very low resolution face recognition problem," *IEEE Transactions on image processing*, vol. 21, no. 1, pp. 327–340, 2011.

[11] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE transactions on Image Processing*, vol. 15, no. 8, pp. 2226–2238, 2006.

[12] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[13] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[14] W. Shi, J. Caballero, F. Huszár, 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," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[15] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[16] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4799–4807.

[17] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2472–2481.

[18] J. Li, F. Fang, K. Mei, and G. Zhang, "Multi-scale residual network for image super-resolution," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 517–532.

[19] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[20] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[21] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[22] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[23] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[25] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[26] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Advances in neural information processing systems*, 2018, pp. 7816–7827.

[27] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 82–92.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[29] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[30] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[31] J. W. Soh and N. I. Cho, "Lightweight single image super-resolution with multi-scale spatial attention networks," *IEEE Access*, vol. 8, pp. 35 383–35 391, 2020.

[32] C. Wang, Z. Li, and J. Shi, "Lightweight image super-resolution with adaptive weighted learning network," *arXiv preprint arXiv:1904.02358*, 2019.

[33] J.-S. Choi and M. Kim, "A deep convolutional neural network with selection units for super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 154–160.

[34] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 286–301.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[36] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 114–125.

[37] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *British Machine Vision Conference (BMVC)*, 2012.

[38] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.

[39] 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," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 416–423.

[40] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[42] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[44] M. Ahmad, M. Abdullah, H. Moon, S. J. Yoo, and D. Han, "Image classification based on automatic neural architecture search using binary crow search algorithm," *IEEE Access*, vol. 8, pp. 189 891–189 912, 2020.

[45] Y. Guo, Y. Luo, Z. He, J. Huang, and J. Chen, "Hierarchical neural architecture search for single image super-resolution," *IEEE Signal Processing Letters*, vol. 27, pp. 1255–1259, 2020.

[46] Y. Weng, Z. Chen, and T. Zhou, "Improved differentiable neural architecture search for single image super-resolution," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1806–1815, 2021.

[47] J. Y. Ahn and N. I. Cho, "Neural architecture search for image super-resolution using densely constructed search space: Deconas," *arXiv preprint arXiv:2104.09048*, 2021.

[48] G. Li, G. Qian, I. C. Delgadillo, M. Muller, A. Thabet, and B. Ghanem, "Sgas: Sequential greedy architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[50] N. Nguyen and J. M. Chang, "Contrastive self-supervised neural architecture search," *arXiv preprint arXiv:2102.10557*, 2021.

[51] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?" in *European Conference on Computer Vision*. Springer, 2020, pp. 798–813.

[52] S. Kaplan and R. Giryes, "Self-supervised neural architecture search," *arXiv preprint arXiv:2007.01500*, 2020.

[53] X. Dong, L. Liu, K. Musial, and B. Gabrys, "Nats-bench: Benchmarking nas algorithms for architecture topology and size," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[54] M. Lindauer and F. Hutter, "Best practices for scientific research on neural architecture search," *Journal of Machine Learning Research*, vol. 21, no. 243, pp. 1–18, 2020.

[55] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7105–7114.

[56] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[57] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel

dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 965–12 974.

[58] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[60] MATLAB, *version 9.8.0.1298242 (R2020a)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[61] X. Wang, Q. Wang, Y. Zhao, J. Yan, L. Fan, and L. Chen, "Lightweight single-image super-resolution network with attentive auxiliary feature learning," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[62] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[63] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, pp. 4077–4087, 2017.

[64] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[65] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, pp. 3630–3638, 2016.

[66] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[67] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, "Tadam: task dependent adaptive metric for improved few-shot learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 719–729.

[68] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1842–1850.

[69] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," in *International Conference on Learning Representations*, 2018.

[70] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," in *International Conference on Learning Representations*, 2018.

[71] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layerwise metric and subspace," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2927–2936.

# 초 록

이미지 복원은 다양한 영상처리 문제의 성능을 높이기 위한 전 처리 단계로 사용할 수 있는 중요한 기술이다. 이미지 고해상도화는 이미지 복원방법 중 중요한 문제의 하나로써 저해상도의 이미지를 고해상도의 이미지로 복원하는 방법이다. 최근에는 컨벌루션 신경망 (CNN)을 사용하는 딥 러닝(deep learning) 기반의 방법들이 단일 이미지 고해상도화 (SISR) 문제를 푸는데 많이 사용되고 있다. 일반적으로 이미지 고해상도화 성능은 CNN을 깊게 쌓거나 복잡한 구조를 설계함으로써 향상시킬 수 있다.

그러나 주어진 문제에 대한 최적의 구조를 찾는 것은 해당 분야의 전문가라 할지라도 어렵고 시간이 오래 걸리는 작업이다. 이러한 이유로 신경망 구축 절차를 자동화하는 신경망 구조 검색 (NAS) 방법이 도입되었다. 이 논문에서는 신경망 구조 검색 (NAS) 방법을 사용하여 새로운 단일 이미지 고해상도화 방법을 제안하였다.

이 논문에서 제안한 방법은 크게 세 가지로 요약 할 수 있다. 이는 효율적인 신경망 검색기법(ENAS)을 이용한 이미지 고해상도화, 병렬 신경망 검색 기법을 이용한 이미지 고해상도화, 메타 전송 학습을 이용하는 신경망 검색기법을 통한 이미지 고해상도화 이다. 우선, 우리는 주로 영상 분류에 쓰이던 신경망 검색 기법을 영상 고해상도화에 적용하였으며, DeCoNASNet이라 명명된 신경망 구조를 설계하였다. 또한 계층적 검색 전략을 사용하여 지역/전역 피쳐(feature) 합병을 위한 최상의 연결 방법을 검색하였다. 이 과정에서 필요 변수가 적으면서 좋은 성능을 낼 수 있도록 복잡성 기반 페널티 (complexity-based penalty) 를 정의하고 이를 REINFORCE 알고리즘의 보상 신호에 추가하였다. 실험 결과 DeCoNASNet은 기존의 사람이 직접

설계한 신경망과 신경망 검색 기법을 기반으로 설계된 최근의 고해상도화 구조의 성능을 능가하는 것을 확인 할 수 있었다.

우리는 또한 여러 크기의 피쳐(feature)를 학습하기 위해 신경망 검색 기법의 검색 공간을 확대하여 병렬 신경망을 설계하는 방법을 제안하였다. 이 때, 병렬신경망의 각 위치에서 매개 변수를 공유할 수 있도록 하여 병렬신경망의 각 구조끼리 정보를 공유하고 전체 구조를 설계하는데 필요한 매개 변수를 줄이도록 하였다. 실험 결과 제안된 방법을 통해 매개 변수 크기 대비 성능이 좋은 신경망 구조를 찾을 수 있었다. 실험 결과를 통해 확장된 검색 공간에서 여러 크기의 피쳐 (feature)를 학습하였기 때문에 이전 방법보다 복잡한 영역을 더 잘 복원하는 것을 확인하였다.

마지막으로 메타 전송 학습(meta-transfer learning)을 신경망 검색에 적용하여 다양한 크기의 이미지 고해상도화 문제를 해결하는 방법을 제안하였다. 이 논문에서는 메타 전송 학습 방법을 통해 제어기가 여러 크기의 좋은 신경망 구조를 동시에 찾을 수 있도록 설계하였다. 또한 메타 훈련된 신경망 구조는 최종 성능 평가 시 학습의 시작점으로 재사용 되어 최종 이미지 고해상도화 성능을 더욱 향상시킬 수 있었으며, 효과적으로 검색-평가 괴리 문제를 해결하였다.

**주요어**: 신경망 검색기법, 이미지 고해상도화, 이미지 복원, 합성곱 신경망
**학번**: 2014-22565