Data-Free Knowledge Distillation For Image Super-Resolution

Yiman Zhang[1], Hanting Chen[1,3], Xinghao Chen[1], Yiping Deng[2], Chunjing Xu[1], Yunhe Wang[1*]
[1] Noah's Ark Lab, Huawei Technologies. [2] Central Software Institution, Huawei Technologies.
[3] Key Lab of Machine Perception (MOE), Dept. of Machine Intelligence, Peking University.
{yiman.zhang, chenhanting, yunhe.wang}@huawei.com

Abstract

Convolutional network compression methods require training data for achieving acceptable results, but training data is routinely unavailable due to some privacy and transmission limitations. Therefore, recent works focus on learning efficient networks without original training data, i.e.*, data-free model compression. Wherein, most of existing algorithms are developed for image recognition or segmentation tasks. In this paper, we study the data-free compression approach for single image super-resolution (SISR) task which is widely used in mobile phones and smart cameras. Specifically, we analyze the relationship between the outputs and inputs from the pre-trained network and explore a generator with a series of loss functions for maximally capturing useful information. The generator is then trained for synthesizing training samples which have similar distribution to that of the original data. To further alleviate the training difficulty of the student network using only the synthetic data, we introduce a progressive distillation scheme. Experiments on various datasets and architectures demonstrate that the proposed method is able to be utilized for effectively learning portable student networks without the original data,* e.g.*, with 0.16*dB *PSNR drop on Set5 for* $\times 2$ *super resolution. Code will be available at https://github.com/huawei-noah/Data-Efficient-Model-Compression.*

1. Introduction

Deep convolutional neural networks have achieved huge success in various computer vision tasks, such as image recognition [12], object detection [26], semantic segmentation [27] and super-resolution [7]. Such great progress largely relies on the advances of computing power and storage capacity in modern equipments. For example, ResNet-50 [12] requires a $\sim$98*MB* storage and $\sim$4*G* FLOPs. However, due to the heavy computation cost of these deep models, they cannot be directly embedded into mobile devices with limited computing capacity, such as self-driving cars, micro-robots and cellphones. Therefore, how to compress CNNs with enormous parameters and then apply them on resource-constrained devices, becomes a research hotspot.

In order to accelerate the pre-trained heavy convolutional networks, various attempts have been made recently, including quantization [24], NAS [5, 39], pruning [25, 34], knowledge distillation [37, 38] and etc. For example, Han *et al*. [11] utilize pruning, quantization and Huffman coding to compress a deep model with extremely higher compression and speed-up ratio. Hinton *et al*. [14] propose knowledge distillation, which learns a portable student network from a heavy teacher network. Luo *et al*. [29] propose ThiNet to perform filter pruning by solving an optimization problem. Courbariaux *et al*. [6] propose binary neural network with only binary weights and activations to extremely reduce the networks' computation cost and storage consumption. Han *et al*. [10] introduce cheap operations in GhostNet to generate more features for lightweight convolutional models.

Although the compressed models with low computation complexity can be easily deployed in mobile devices, these techniques require original data to fine-tune or train the compressed networks to achieve comparable performance with the pre-trained model. However, the original training data is often unavailable due to some privacy or transmission constraints. For example, Google shared a series of excellent models trained on the JFT-300M dataset [20] which is still unpublic. In addition, there are considerable apps trained using privacy-related data such as face ID and voice assistant, which is often not provided. It is very hard to provide model compression and acceleration service for these models without the original training data. Therefore, existing network compression methods cannot be well performed.

To this end, recent works are devoted to compress and accelerate the pre-trained heavy deep models without the training dataset, *i.e*. data-free model compression. Lopes *et al*. [28] first propose to use meta data to reconstruct the orig-

*Corresponding author.

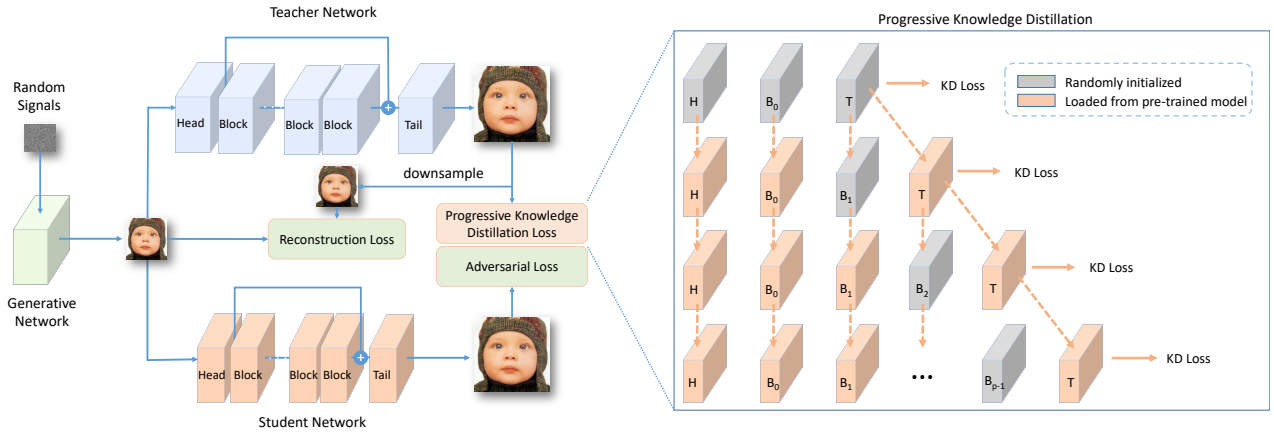

Figure 1. Framework of the proposed data-free knowledge distillation. The generator is trained with reconstruction loss and adversarial loss to synthesize images that similar with the original data. The student network is then obtained utilizing progressive distillation from the teacher network.

inal dataset for knowledge distillation. Nayak *et al*. [32] present zero-shot knowledge distillation, which leverages the information of pre-trained model to synthesize useful training data. Chen *et al*. [4] exploit Generative Adversarial Networks (GAN) to generate training samples which have similar distribution with the original images and achieve better performance. Yin *et al*. [40] propose DeepInversion and successfully generate training data on the ImageNet dataset. However, they only focus on image recognition tasks with sophisticated loss function, *e.g*., the one-hot loss for capturing features learned by the conventional cross-entropy loss. Differently, the training of SISR models does not involve the semantic information, the ground-truth images are exactly the original high-resolution images. Thus, the existing data-free compression approaches cannot be directly employed.

To this end, we propose a new data-free knowledge distillation framework for super-resolution. A generator network is also adopted for approximating the original training data from the given teacher network. Different from the classification networks whose outputs are probability distributions, the inputs and outputs of SR models are images with similar patterns. Therefore, we develop the reconstruction loss function by utilizing this relationship. In practice, we have to ensure that the synthetic images will not be distorted significantly by the teacher SR network, *i.e*., the super-resolution results of these images should be similar to themselves. Moreover, an adversarial loss is combined to prevent the model collapse of the generator. Since SISR models are often required to capture and emphasize details such as edge and texture from the input images, the learning on intermediate features is also very important. Thus, we propose to conduct the distillation progressively to alleviate the training difficulty. We then conduct a series of experiments on several benchmark datasets and models. The results demonstrate that the proposed framework can effectively learn a portable network from a pre-trained model without any training data.

The rest of this paper is organized as follows. Section 2 investigates related work about model compression in super-resolution and data-free knowledge distillation methods. Section 3 introduces our data-free distillation method for image super-resolution. Section 4 provides experimental results on several benchmark datasets and models and Section 5 concludes the paper.

2. Related Work

In this section, we briefly review the related works on data-driven knowledge distillation for super-resolution networks and data-free model compression approaches.

2.1. Data-Driven Knowledge Distillation for Super-Resolution

As there are urgent demands for applying image super-resolution networks to the mobile devices such as cellphones and cameras, various attempts have been made to learn lightweight super-resolution models.

Gao *et al*. [9] calculate different statistical maps of feature maps to distill from teacher super-resolution networks. He *et al*. [13] propose a feature affinity-based knowledge distillation (FAKD) framework for super-resolution networks, which improves the distillation performance by using the correlation within a feature map. Lee *et al*. [22] propose to utilize ground truth high resolution images as privileged information and use feature distillation to improve the performance of compact super-resolution network. To enhance the performance of lightweight super-resolution networks, Zhang *et al*. [42] introduce the con-

cept of learnable pixel-wise importance map, and use prediction of teacher network to initialize the importance map. In addition, Hui *et al*. [16] and Jiang *et al*. [17] design new structures to perform distillation between different parts of the model and improve the performance of the lightweight super-resolution network.

The compressed models obtained by the aforementioned methods can achieve promising performance with low computation cost. However, launching these techniques requires the original training dataset. In practice, the training dataset is often unavailable due to the privacy or transmission concerns. Therefore, it is essential to explore the data-free model compression methods.

2.2. Data-Free Model Compression

There are lot of works focusing on data-free model compression, *i.e*., learning a portable network without any training data. Lopes *et al*. [28] realize a data-free knowledge distillation method for classification by generating samples only based on some meta data stored during distillation. Similarly, Bhardwaj *et al*. [3] calculate centroids of classes by using part of real dataset and then generate distillation data to distill from teacher network. Both these methods exploit meta data of training datasets and iterate to update the input noise images to synthesize samples for distillation.

There are also some methods which propose a purely data-free distillation framework, *i.e*., without any training datasets or meta data. Nayak *et al*. [32] model the output of teacher network as a dirichlet distribution and iterate the input noise images to obtain training samples. This method performs well in MNIST, but gets more than 10% accuracy drop in Fashion MNIST [36] and CIFAR-10 [21] when compared with baselines that were trained with original data.

Different from optimizing noises images iteratively, Chen *et al*. [4] exploit GAN to generate training samples by optimizing the parameters of the generator network via the customized one-hot loss, information loss and activation loss based on characteristics of classification. This data-free learning method (DAFL) provides a novel framework and gets less than 5% accuracy drop in CIFAR-10 and CIFAR-100. Addepalli *et al*. [1] propose DeGAN, which uses generator to synthesize images and utilizes test data to improve the quality of images. Similarly, Paul *et al*. [31] and Fang *et al*. [8] also use generator to generate images for training. Unlike DAFL [4] which treats generation and distillation as separate procedures, these methods consider them as two competing process, forcing generator to produce images to make student's outputs differ from teacher while the student is trained to imitate teacher network. Their methods perform well both in classification and segmentation tasks.

Yin *et al*. [40] propose a method that combines updating noise images and adversarial loss between teacher and student for classification tasks. In addition, based on the observation that Batch Normalization (BN) layers which contain channel-wise means and variances of training dataset are widely used in classification neural networks, they propose to use these BN data to inverse teacher network and generate samples. This method provides comparable performance in classification and generates samples that are very similar to natural images.

Although these methods can successfully compress networks without any data, most of them focus on image recognition tasks and design specific loss functions to inherit useful information from the pre-trained networks, which cannot be transferred to other tasks such as image super-resolution.

3. Data-Free Learning for Super-resolution

To provide better compressed models while protecting user privacy, we propose a data-free knowledge distillation framework for super-resolution networks.

3.1. Training Samples Generation

Let $\mathcal{T}$ be a pre-trained teacher super-resolution network, which has desired super-resolution visual effect but is difficult to be used in mobile devices. Let $\mathcal{S}$ be the student network, which has desired speed in mobile devices and its structure can be customized based on hardware capabilities.

By applying knowledge distillation, the student network is trained by the following function:

$$\mathcal{L}_{KD} = E_{x \in p_x(x)}[\|\mathcal{T}(x) - \mathcal{S}(x)\|_1], \tag{1}$$

where x is the training sample and $p_x(x)$ is the distribution of the original dataset. However, we cannot train the student network $\mathcal{S}$ when the original training sample x is not available.

To adjust this problem, recent works [4, 8] propose to utilize a generator to synthesize training samples. However, these methods design special loss function to capture the distribution of training data in classification networks, which cannot be applied in image super-resolution tasks. Thus, we want to use the fundamental characteristic of super-resolution networks to help generator produce images similar with the real images. In super-resolution tasks, models take low-resolution images as input and output high-resolution images. Compared with low-resolution images, high-resolution images have more pixels and add more details while preserving all information of low-resolution images. In other words, if we down-sample a high-resolution image to the size same as its corresponding low-resolution image, the two images should be the same theoretically.

Specifically, given a low-resolution image $I_L \in \mathcal{R}^{H \times W \times C}$ and a pre-trained super-resolution model, we can get its super-resolution result $I_S \in \mathcal{R}^{sH \times sW \times C}$. Then

Algorithm 1 Algorithm with Progressive Distillation

Input: A pre-trained super-resolution teacher model $\mathcal{T}$; P indicates the number of student body segments; M denotes batchsize; $p(z)$ denotes noise prior.

Initialize: Randomly initialize a student model $\mathcal{S}(x;\theta^s)$ and a generator $\mathcal{G}(x;\theta^g)$
Initialize the Set $\{S_i(x;\theta^s)\}_{0\leq i<P, i\in\mathbb{N}^+}$ based on $\mathcal{S}(x;\theta^s)$ and $\mathcal{G}(x;\theta^g)$ randomly.
for $k = 0$ *to* P **do**
  Initialize $\mathcal{S}_k \leftarrow \mathcal{S}_{max(k-1.0)}$.
  for number of training iterations **do**
    Imitation Stage:
    for k steps **do**
      Sample noise images $\{z^i\}_{i\leq M}$ from $p(z)$.
      Get generated images $\{\mathcal{G}(z^i)\} \leftarrow \{z^i\}$.
      Obtain SR results $\{\mathcal{T}(\mathcal{G}(z^i))\}, \{\mathcal{S}_k(\mathcal{G}(z^i))\}$.
      Calculate loss $\mathcal{L}_{S_k}$ via Eq. (5).
      Update θ^{s_k} with $\nabla\mathcal{L}_{S_k}$.
    end for
    Generation Stage:
    Repeat step $8 \sim 10$.
    Calculate loss $\mathcal{L}_G$ with (4).
    Update θ^g with $\nabla\mathcal{L}_G$.
  end for
end for

Output: Output the trained student network $\mathcal{S}(x;\theta^s)$.

we can scale the high-resolution image I_S to the low-resolution size and get $I_{SL} \in \mathcal{R}^{H\times W\times C}$. H, W, C indicate the height, the width and the channel of I_L respectively. s indicates the super-resolution scale. Considering that low-resolution images are generally obtained by the interpolation of high-resolution images in most super-resolution network training process, while given a well-trained super-resolution model, we hold that I_{SL} and I_L should be consistent when I_L is a natural dataset image. To distill from teacher super-resolution network efficiently, generator $\mathcal{G}$ is expected to produce samples which follow the distribution of the dataset.

Denote $\mathcal{G}$ as the generator to produce training samples, given a random variable z from a distribution p_z as input, the image synthesized by the generator network is $\mathcal{G}(z)$. The super-resolution result of $\mathcal{G}(z)$ using teacher network $\mathcal{T}$ is $\mathcal{T}(\mathcal{G}(z))$. Then we rescale $\mathcal{T}(\mathcal{G}(z))$ to the size of $\mathcal{G}(z)$ and get $R(\mathcal{T}(\mathcal{G}(z)))$. Generator $\mathcal{G}$ is expected to produce samples which follow the distribution of the dataset and for a dataset image I_L, its I_{SL} stays consistent with itself, then $R(\mathcal{T}(\mathcal{G}(z)))$ should be consistent with $\mathcal{G}(z)$. Therefore we propose a reconstruction loss for the generator, which is formulated as Eq. (2):

$$\mathcal{L}_R = E_{z\in p_z(z)}[\frac{1}{n}\left\|R(\mathcal{T}(\mathcal{G}(z))) - \mathcal{G}(z)\right\|_1], \quad (2)$$

where $R(\cdot)$ denotes $1/s$ times interpolation.

However, directly applying Eq. (2) to the generator may converge to a trivial solution: the generator will produce a single image which satisfies this loss function. In this situation, the generator cannot synthesize a set of different samples for distilling the student network. Therefore, we introduce the adversarial loss to maintain the diversity of training samples.

Inspired by DFAD [8], we use the adversarial loss to distill from teacher super-resolution networks without access to the original dataset or any related datasets. The generator network is optimized to produce hard samples to maximize the model discrepancy between teacher and student. The adversarial loss $\mathcal{L}_{GEN}$ is formulated as:

$$\mathcal{L}_{GEN} = -log(\mathcal{L}_{KD} + 1), \quad (3)$$

where the log function is used to slow down the training of the generator and make training more stable. Therefore, the loss function to optimize the generator can be formulated as:

$$\mathcal{L}_G = \mathcal{L}_{GEN} + w_R\mathcal{L}_R, \quad (4)$$

where w_R is the trade-off hyper-parameter to balance the two terms.

3.2. Progressive Distillation

While training student by distilling from teacher without true data, we expect the output of generator to obey the distribution of training datasets. In the scenario where no data is available, it's difficult to train student network directly by distilling information from teacher network. Considering that most super resolution networks have many blocks or layers, to better train the student network, we propose to train a tiny network firstly. This tiny network has similar structure with student network but less parameters to be trained. Thus it is much easier for the tiny network to be optimized. Then with the trained tiny network, we increase the number of layers or blocks gradually and train the parameters in those new layers or blocks sequentially. With this progressive distillation method we can distill more information from teacher network and train student network better.

More specifically, for a super resolution network $\mathcal{S}$, its function can be formulated as $\mathcal{S}(x) = \mathcal{S}_T(\mathcal{S}_B(\mathcal{S}_H(x)))$, where x indicates the input of $\mathcal{S}$, $\mathcal{S}_H$, $\mathcal{S}_B$ and $\mathcal{S}_T$ indicate the head, body and tail of $\mathcal{S}$ respectively. Given that $\mathcal{S}_B$ contains N layers or blocks, we can split it into P parts $\{B_i\}_{0\leq i<P, i\in\mathbb{N}^+}$ and train $\{B_i\}_{0\leq i<P, i\in\mathbb{N}^+}$ in P steps. Initially, based on $\mathcal{S}_H$, $\mathcal{S}_T$ and B_0, we build a network $\mathcal{S}_0$ and initialize it randomly. The function of $\mathcal{S}_0$ can be formulated as $\mathcal{S}_0(x) = \mathcal{S}_T(B_0(\mathcal{S}_H(x)))$. In the process of training $\mathcal{S}_0$, the knowledge distillation loss is $\mathcal{S}_0$ formulated as:

$$\mathcal{L}_{KD_{S_i}} = E_{z\in p_z(z)}[\frac{1}{n}\left\|\mathcal{T}(G(z)) - \mathcal{S}_i(G(z))\right\|_1], \quad (5)$$

Table 1. Quantitative results (PSNR/SSIM) of VDSR in different experimental settings.

Dataset	Scale	VDSR									
		Teacher		Student		Bicubic		Noise		**Ours**	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set5	×2	37.59	0.9597	37.36	0.9588	33.69	0.9308	34.38	0.9376	37.20	0.9582
	×3	33.75	0.9232	33.50	0.9209	30.41	0.8692	30.53	0.8711	33.13	0.9172
	×4	31.40	0.8854	31.17	0.8819	28.43	0.8114	28.46	0.8120	30.71	0.8728
Set14	×2	33.16	0.9140	32.97	0.9127	30.34	0.8701	30.81	0.8821	32.84	0.9109
	×3	29.92	0.8343	29.81	0.8323	27.64	0.7757	27.74	0.7780	29.63	0.8291
	×4	28.14	0.7706	28.01	0.7675	26.10	0.7044	26.12	0.7050	27.73	0.7617
B100	×2	31.91	0.8964	31.79	0.8950	29.56	0.8437	30.07	0.8603	31.56	0.8920
	×3	28.84	0.7983	28.74	0.7961	27.21	0.7391	27.27	0.7417	28.56	0.7920
	×4	27.28	0.7256	27.21	0.7234	25.96	0.6680	25.97	0.6686	27.03	0.7187
Urban100	×2	30.79	0.9147	30.48	0.9111	26.88	0.8407	27.27	0.8523	29.74	0.9017
	×3	27.16	0.8291	26.92	0.8229	24.46	0.7351	24.55	0.7380	26.37	0.8081
	×4	25.20	0.7533	25.01	0.7467	23.14	0.6577	23.16	0.6584	24.54	0.7298

where $0 \leq i<P, i \in \mathbb{N}$. After training $\mathcal{S}_0$ for several steps, we add B_1 into $\mathcal{S}_0$ and get $\mathcal{S}_1$, which performs as $\mathcal{S}_1 = \mathcal{S}_T(B_1(B_0(\mathcal{S}_H(x))))$. Then when training $\mathcal{S}_1$, we initialize $\mathcal{S}_1$ with trained $\mathcal{S}_0$ and use the strategy of training $\mathcal{S}_0$ to train $\mathcal{S}_1$. We repeat this training process until $\mathcal{S}_{P-1}$ is trained. The architectures in different steps and the way its initialized are shown in Figure 1.

3.3. Optimization

By combining all the aforementioned loss functions and the progressive distillation method, we obtain the final objective functions Eq. (4) for generator and Eq. (6) for student respectively.

$$\mathcal{L}_S = \mathcal{L}_{KD_{S_{P-1}}}. \tag{6}$$

Our training strategy is summarized as Algorithm. 1. We apply an iterative and progressive training strategy to optimize generator $\mathcal{G}$ and student network $\mathcal{S}$. While given student network $\mathcal{S}$ and the number of segments of student network body P, we construct a set of tiny student networks $\{S_i(x;\theta^s)\}_{0\leq i<P, i\in\mathbb{N}^+}$ as discussed in the Section 3.2. Then we train $\{S_i\}$ in turn and initialize S_{i+1} with trained S_i. The training process of S_i and $\mathcal{G}$ are performed alternately. In one iteration, we fix the generator $\mathcal{G}$, calculate knowledge distillation loss with Eq. (5) and then update the parameter of S_i via backward propagation. After updating S_i for several steps, we fix the parameter of S_i and calculate Eq. (4) to optimize $\mathcal{G}$. It's worth noting that when we start training S_{i+1}, the generator $\mathcal{G}$ will not be reinitialized. After all the networks in $\{S_i(x;\theta^s)\}_{0\leq i<P, i\in\mathbb{N}^+}$ are trained according to the preceding procedure, the training process of our student network $\mathcal{S}$ is complete.

4. Experiments

In this section, we conduct extensive experiments to verify the effectiveness of the proposed data-free distillation method on various super-resolution datasets. Quantitative and qualitative results are compared with baselines of VDSR [18] and EDSR [23].

4.1. Baselines

A bunch of baselines is compared to demonstrate the effectiveness of our proposed method. The baselines are briefly described as follows.

Teacher: the given pre-trained model which serves as the teacher in the distillation process.

Student: the student trained with original training data.

Noise: the student trained with randomly generated noise images.

4.2. Experiments on VDSR

Firstly, we experiment with our method on VDSR [18]. We choose VDSR model as the teacher super-resolution model, and then halve the number of channels in teacher network to get our student network (denoted as VDSR-half). Same with [18], we use 291 images as in [33] for training and Set5 for validation in our experiments. Besides, we make some modifications to the generator in [4] and use it as our generator. Specifically, we remove the last BN layer and replace the remaining BN layers with instance normalization (IN) layers.

Our method is implemented based on the open source Pytorch code of VDSR[1] and experiments are conducted on a NVIDIA V100 GPU. The optimizers for student and generator are SGD and Adam, respectively. Different from

[1] https://github.com/twtygqyy/pytorch-vdsr

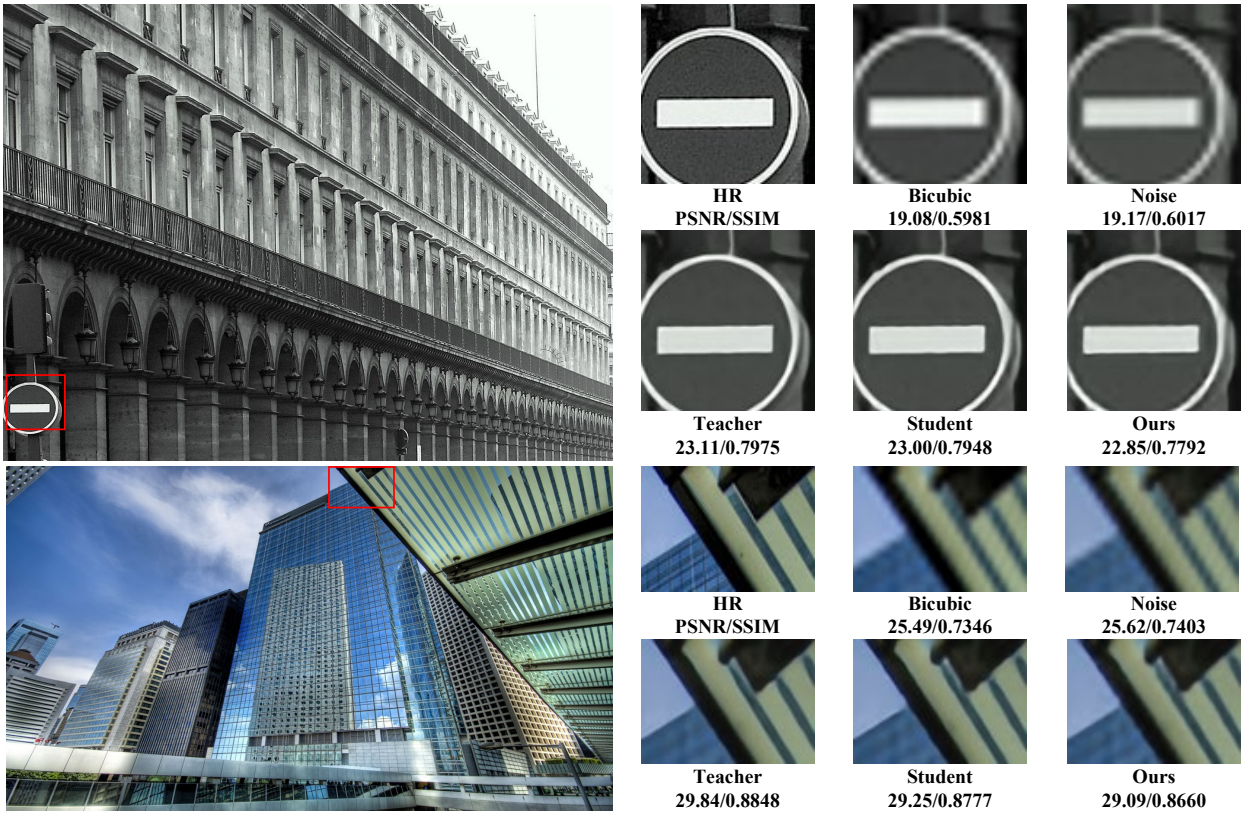

Figure 2. $\times 4$ super resolution results of img066 and img061 from Urban100 on VDSR.

41×41 images used for training in VDSR [18], we process 48×48 images instead to distill a single student model for multiple scales. In particular, we use 3 generators corresponding to 3 super resolution scale to generate images for distillation. The output size of each generator is calculated as $r = 48/scale$. At each update of training, we randomly select a scale among $\times 2$, $\times 3$ and $\times 4$. Only the generator that corresponds to the selected scale is enabled to construct the mini-batch and updated. The student network and all generators are randomly initialized. During the optimization, for student network, the learning rate is initially set to 0.1 and then decreased by a factor of 10 every 10 epochs. For generator, the learning rate is initially set to 1e-5 and attenuated according to the same policy as that of student network. Additionally, throughout the training process, we process 120 iteration each epoch and in each iteration, we first update student network 50 times then update the generator 1 time. The reconstruction weight w_R is set to 1.0. In addition, VDSR is divided to 2 parts and trained in two stages. The first phase trains 12 epochs and then the total network is trained for 68 epochs.

Table 1 shows the performance of the student model obtained with different methods. In this table, Teacher indicates the pre-trained teacher model, Student indicates the student model that is trained on original dataset, and Noise indicates that we use images random sampled from uniform distribution for distillation. As is shown in the table, our method performs significantly better than training with random noise images, and achieves results close to training with original dataset. For example, the student model trained with our method gets only $0.16dB$ decrease on Set5 for scale 2 compared with training with original dataset. The visual qualities of the same architecture using different training strategies are shown in Figure 2. Our method shows similar visual quality with student trained with original dataset and performs better than training with noise images and bicubic results.

In Figure 3, we visualize several images that are synthesized by generators and their corresponding super resolution results by teacher network. Considering that the channel number of data processed by VDSR is single, we concatenate three synthetic images to obtain better visual effects. The corresponding SR results are also concatenated together.

4.3. Ablation Experiments

Here we will conduct the detailed ablation study to illustrate the effectiveness of different components in our pro-

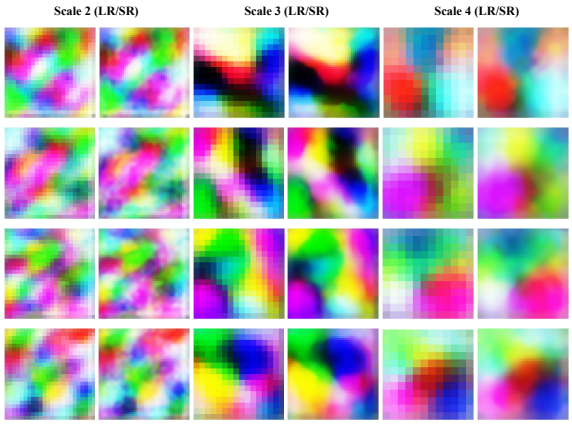

Figure 3. Samples of images that synthesized by generators.

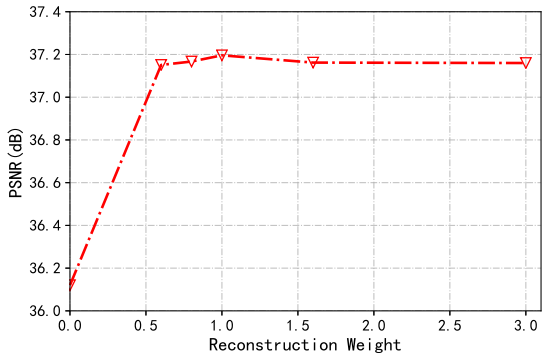

Figure 4. The influence of reconstruction weight on Set5 $\times 2$.

posed method. Results are shown in Table 2.

The ablation experiments are also conducted on VDSR. We use VDSR as a teacher network and VDSR-half as a student network. The training settings are the same as those in Section 4.2. Table 2 reports the results of various design components for scale 2. M1 denotes distillation with images sampled from random uniform distribution and achieves PSNR of only $34.38dB$. When further training the generator with the reconstruction loss, we observe $2.67dB$ PSNR improvement. If we train the generator with DFAD[8], the PSNR is $36.23dB$. Combining the adversarial loss and reconstruction loss (M3) results in significant better performance with $37.09dB$ PSNR. The results of M4 and M2 illustrate the effectiveness of the proposed reconstruction loss. Further applying progressive distillation method brings in another $0.11dB$ increase, which demonstrates that the proposed progressive distillation can further improve performance.

Then we examine the impact of the weight for reconstruction loss, *i.e.* w_R. As is shown in Figure 4, while increasing w_R from 0 to 3, the PSNR on Set5 for $\times 2$ increases, then slightly decreases, and then remains stable. When the weight is set to 1, we get the best results.

Table 2. Effectiveness of different components of the proposed data-free learning method on VDSR for scale 2. AD means adversarial loss, RS indicates reconstruction loss and PD is progressive distillation.

Model	AD	RS	PD	Set5
M1	×	×	×	34.38
M2	×	√	×	37.05
DFAD[8]	√	×	×	36.23
M3	√	√	×	37.09
M4	√	×	√	36.12
Ours	√	√	√	37.20

Table 3. Model specifications in EDSR.

Options	Student	Teacher
Residual Blocks	32	32
Filters	128	256
Residual Scaling	0.1	0.1

4.4. Experiments on EDSR

We also conduct experiments on EDSR, which is proposed by Lin *et al*. [23] in 2017 and widely used in super-resolution community. We choose EDSR models as our teacher super-resolution models and tiny models with half channels (EDSR-half) as our student models. Following the setting in EDSR, our teacher models contain 32 residual blocks, 256 filters and residual scaling is set to 0.1. Our student super-resolution models contain 32 residual blocks, 128 filters and same residual scaling with teacher model. Details are shown in Table 3. Moreover, the generator used here is same with VDSR.

Hyper-parameters that used in training teacher super-resolution models are totally the same as those in EDSR. The teacher super-resolution models are trained on DIV2K dataset [35]. DIV2K dataset [35] consists of 800 2K resolution training images and 100 2K resolution validation images and is widely used for training and testing in super-resolution tasks. To evaluate our proposed method, we calculate PSNR and SSIM on Set5 [2], Set14 [41], B100 [30] and Urban100 [15].

Based on the open source code of EDSR [23], our method is implemented with Pytorch on an NVIDIA V100 GPU. During training student network and generator network iteratively, we process 16 48×48 images per step. For the training process of student network, we set the learning rate as 10^{-4} and optimize the parameter with ADAM optimizer [19] by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon_1 = 0.9$. For training the generator network, we set the learning rate as 10^{-6} and also use ADAM as optimizer. We update the student network 50 times and then optimize the generator network. During the whole training process, we optimize for 300 epochs and in every epoch, we update generator

Table 4. Quantitative results (PSNR/SSIM) in different experimental settings on EDSR.

Dataset	Scale	EDSR									
		Teacher		Student		Bicubic		Noise		**Ours**	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set5	×2	38.19	0.9609	38.15	0.9610	33.69	0.9308	34.65	0.9425	37.67	0.9593
	×3	34.68	0.9294	34.54	0.9283	30.41	0.8692	30.22	0.8448	33.57	0.9203
	×4	32.48	0.8988	32.32	0.8968	28.43	0.8114	28.39	0.8021	31.78	0.8897
Set14	×2	33.95	0.9202	33.78	0.9197	30.34	0.8701	31.35	0.8960	33.16	0.9138
	×3	30.53	0.8465	30.45	0.8447	27.64	0.7757	27.86	0.7776	29.77	0.8332
	×4	28.82	0.7879	28.72	0.7852	26.10	0.7044	26.27	0.7170	28.33	0.7758
B100	×2	32.35	0.9109	32.26	0.9008	29.56	0.8437	30.32	0.8753	31.76	0.8942
	×3	29.26	0.8096	29.18	0.8074	27.21	0.7391	27.09	0.7271	28.66	0.7945
	×4	27.72	0.7420	27.65	0.7391	25.96	0.6680	25.88	0.6772	27.38	0.7292
Urban100	×2	32.97	0.9359	32.50	0.9321	26.88	0.8407	28.56	0.8819	30.58	0.9119
	×3	28.81	0.8659	28.50	0.8596	24.46	0.7351	24.64	0.7273	26.74	0.8195
	×4	26.65	0.8036	26.38	0.7958	23.14	0.6577	23.40	0.6670	25.40	0.7609

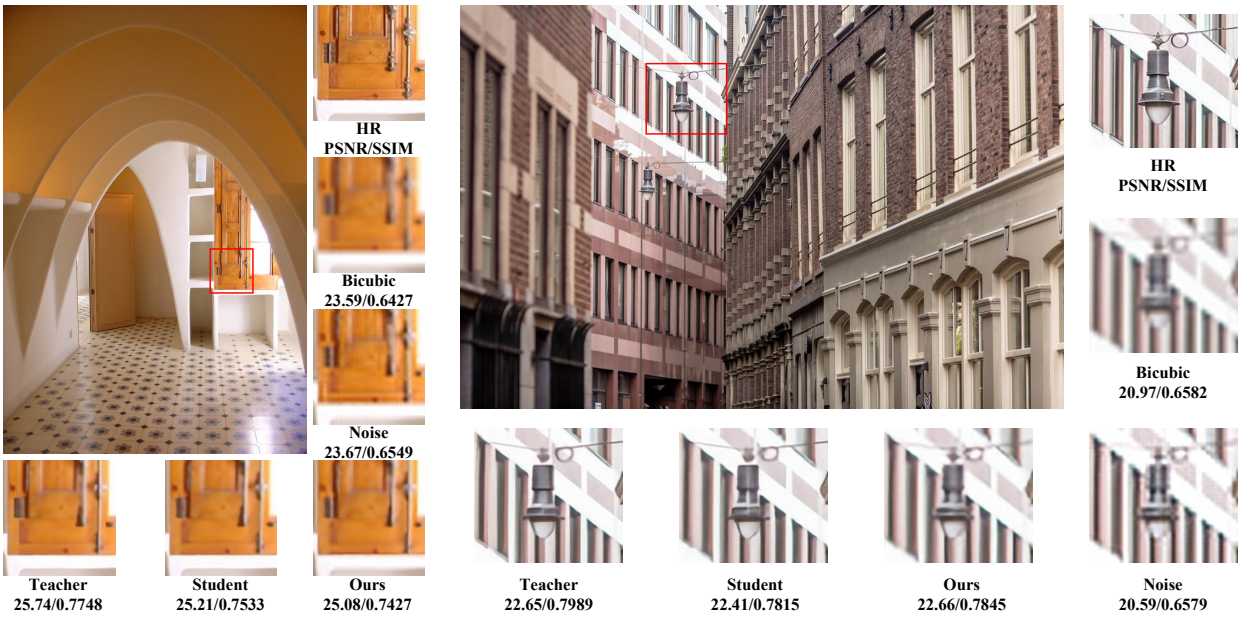

Figure 5. ×4 super resolution results of img080 and img064 from Urban100 on EDSR.

120 times. Same with VDSR, we set w_R to 1.0. In addition, the student network is divided in 2 parts and trained for 2 phases. In the first phase, we train 80 epochs.

The performance of our method and the compared baselines are provided in Table 4. As is shown in Table 4, our proposed method performs well in EDSR. For instance, for ×2 super resolution, our method gets about $3.05dB$ increase than distilling with random noise images and is only $0.48dB$ lower than training with datasets.

5. Conclusion

This paper studies the data-free model compression for single image super-resolution networks, which are widely used in portable devices. We first analyze the difference between the mechanisms of image classification and super-resolution. Then, we propose to utilize the reconstruction loss and the adversarial loss to train the generator for approximating the original training data. Moreover, a progressive distillation strategy is imposed on the student network for better inheriting useful information from teacher network. Extensive experiments demonstrate that our proposed method can produce student networks with similar results and fewer computational costs without training data. In addition, the proposed method can be easily transferred to other low-level computer vision tasks such as image denoising and inpanting, which will be investigated in future works.

References

[1] Sravanti Addepalli, Gaurav Kumar Nayak, and Anirban Chakraborty. Degan: Data-enriching gan for retrieving representative samples from a trained classifier.

[2] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

[3] Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework. *arXiv preprint arXiv:1905.07072*, 2019.

[4] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3514–3522, 2019.

[5] Hanlin Chen, Baochang Zhang, Xiawu Zheng, Jianzhuang Liu, David Doermann, Rongrong Ji, et al. Binarized neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10526–10533, 2020.

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[8] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation, 2020.

[9] Qinquan Gao, Yan Zhao, Gen Li, and Tong Tong. Image super-resolution using knowledge distillation. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 527–541, Cham, 2019. Springer International Publishing.

[10] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Zibin He, Tao Dai, Jian Lu, Yong Jiang, and Shu-Tao Xia. Fakd: Feature-affinity based knowledge distillation for efficient image super-resolution. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 518–522. IEEE, 2020.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2015.

[16] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 723–731, 2018.

[17] Kui Jiang, Zhongyuan Wang, Peng Yi, Junjun Jiang, Jing Xiao, and Yuan Yao. Deep distillation recursive network for remote sensing imagery super-resolution. *Remote Sensing*, 10(11):1700, 2018.

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2019.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Wonkyung Lee, Junghyup Lee, Dohyung Kim, and Bumsub Ham. Learning with privileged information for efficient image super-resolution. *arXiv preprint arXiv:2007.07524*, 2020.

[23] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[24] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7474–7485. Curran Associates, Inc., 2020.

[25] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*, 2020.

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[28] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[29] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *CVPR*, pages 5058–5066, 2017.

[30] David Martin, Charless Fowlkes, Doron Tal, and Jitendra 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*, volume 2, pages 416–423. IEEE, 2001.

[31] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, pages 9551–9561, 2019.

[32] GK Nayak, KR Mopuri, V Shaj, R Venkatesh Babu, and A Chakraborty. Zero-shot knowledge distillation in deep networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019, pages 8317–8325. International Machine Learning Society (IMLS), 2019.

[33] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015.

[34] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing XU, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10936–10947. Curran Associates, Inc., 2020.

[35] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei 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*, pages 114–125, 2017.

[36] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[37] Yixing Xu, Yunhe Wang, Hanting Chen, Kai Han, Chunjing XU, Dacheng Tao, and Chang Xu. Positive-unlabeled compression on the cloud. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[38] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7074–7083, 2020.

[39] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1829–1838, 2020.

[40] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[41] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer, 2010.

[42] Lei Zhang, Peng Wang, Chunhua Shen, Lingqiao Liu, Wei Wei, Yanning Zhang, and Anton Van Den Hengel. Adaptive importance learning for improving lightweight image super-resolution network. *International Journal of Computer Vision*, 128(2):479–499, 2020.