MemSR: Training Memory-efficient Lightweight Model for Image Super-Resolution

Kailu Wu [1] **Chung-Kuei Lee** [2] **Kaisheng Ma** [1]

Abstract

Methods based on deep neural networks with a massive number of layers and skip-connections have made impressive improvements on single image super-resolution (SISR). The skip-connections in these complex models boost the performance at the cost of a large amount of memory. With the increase of camera resolution from 1 million pixels to 100 million pixels on mobile phones, the memory footprint of these algorithms also increases hundreds of times, which restricts the applicability of these models on memory-limited devices. A plain model consisting of a stack of 3×3 convolutions with ReLU, in contrast, has the highest memory efficiency but poorly performs on super-resolution. This paper aims at calculating a **winning initialization** from a complex teacher network for a plain student network, which can provide performance comparable to complex models. To this end, we convert the teacher model to an equivalent large plain model and derive the plain student's initialization. We further improve the student's performance through initialization-aware feature distillation. Extensive experiments suggest that the proposed method results in a model with a competitive trade-off between accuracy and speed at a much lower memory footprint than other state-of-the-art **lightweight** approaches.

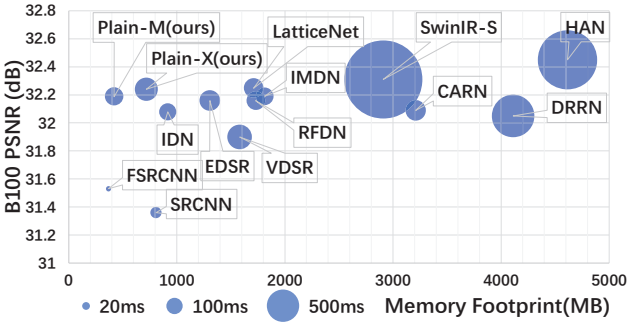

Figure 1. **Speed, accuracy, and memory trade-off.** The maximum memory footprint and the average inference time for upscaling 2× on LR image of size 960 × 540 with full precision on PyTorch (Paszke et al., 2019) and Nvidia GTX 1080Ti are shown above. Accuracy is estimated on B100 (Martin et al., 2001) ×2 dataset in terms of peak signal to-noise ratio (PSNR). The plain model trained with our framework achieves performance comparable to existing light-weighted models while requiring memory comparable to FSRCNN.

1. Introduction

Single image super-resolution (SISR) pursues reconstructing a high-resolution (HR) image from a low-resolution (LR) counterpart, which has been widely applied in mobile phones, televisions, and cameras. With the great success of deep neural networks, the pioneering work SRCNN (Dong et al., 2016a) first used a three-layer convolutional neural network (CNN) for SISR and outperformed conventional approaches by large margins. After that, CNN-based SR methods (Mei et al., 2020; Tai et al., 2017a; Niu et al., 2020; Muqeet et al., 2020) consisting of deep networks with sparse and dense skip-connections achieved impressive performance. However, due to the high computational cost and memory footprint, it is hard to utilize them for real applications such as mobile phones, televisions, and robots. Thus, it is crucial to build lightweight SR models.

Conventional lightweight SR models generally focus on reducing the computational cost or the number of parameters. Therefore, they use global residual learning (Kim et al., 2016a), residual groups (Ledig et al., 2017), residual dense blocks (Zhang et al., 2018b), residual in residual architecture (Zhang et al., 2018a), and channel splits (Hui et al., 2019), which come at the cost of increased memory usage or inference speed. Memory-efficient SR models use recursive layers to reduce parameter counts (Tai et al., 2017a;

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China [2]HiSilicon Technologies, Shanghai, China . Correspondence to: Kaisheng Ma <kaisheng@tsinghua.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Kim et al., 2016a) or adopt additional modules specific for SISR (Ahn et al., 2018; Hui et al., 2019). These specially designed or recursive architectures only reduce parameter numbers and are unfriendly towards hardware (Du et al., 2021). SRCNN (Dong et al., 2016a) and FSRCNN (Dong et al., 2016b) are hardware-friendly and runtime memory-efficient yield poor image quality.

Therefore, it is necessary to train a hardware-friendly network with low memory usage, fast inference speed, and high image quality. However, existing methods can not train a plain model achieving performance comparable to a complicated model. Therefore, we propose MemSR, an effective framework for training a plain network in SISR. Our algorithm transfers the teacher knowledge to the student network by deriving an initialization from the teacher. Here, we name it **winning initialization**.

To this end, we convert the teacher network with multi-branch topology to an equivalent cumbersome plain model and find a winning initialization for a lightweight student from this plain model. Then we further improve the student performance with a specialized feature distillation, where the parameters in distillation are based on the winning initialization of the student.

We mainly exploit EDSR (Lim et al., 2017) as the teacher structure since it is a well-known SISR model supported by our framework. Experimental results demonstrate that the winning initialization improves the plain model performance by a large margin. To the best of our knowledge, our framework is the first attempt to boost plain model performance for SISR. To summarize, we make the following key contributions:

- We present a novel framework that can effectively train a plain model, resulting in a competitive PSNR-speed trade-off compared to state-of-the-art lightweight SR models and favorable memory-PSNR trade-off.
- We propose a distillation method coupled with student initialization, which significantly surpasses conventional SR distillation on plain models.
- We demonstrate the effectiveness of our framework with extensive experiments on different scales of students.

2. Related Work

2.1. Single Image Super-Resolution

With the promising performance achieved by applying deep learning in SISR, methods based on convolutional neural networks (CNNs) have been the mainstream. The pioneering work SRCNN (Dong et al., 2016a) directly applied a three-layer convolutional neural network to the task. VDSR (Kim et al., 2016b) employed skip-connections to learn the residual information between low-resolution images and high-resolution images. EDSR (Lim et al., 2017) improved the SRResNet (Ledig et al., 2017) by removing Batch Normalization (Ioffe & Szegedy, 2015) and dramatically advanced the SR performance. Inspired by DenseNet (Huang et al., 2017), RDN introduced dense connections into SR network structures and improved performance with fewer parameters.

Recent methods apply sparse (Mao et al., 2016; Ledig et al., 2017; Chu et al., 2020) or dense (Haris et al., 2018; Zhang et al., 2018b; Tong et al., 2017) skip connections to prevent the gradient vanishing problem and boost the performance using attention mechanisms (Hu et al., 2018; Zhang et al., 2019; Xiao et al., 2019; Mei et al., 2020; Zhao et al., 2020; Muqeet et al., 2020) or advanced sub-block designs (Luo et al., 2020; Niu et al., 2020). Although these models make significant improvements quantitatively and qualitatively, a massive number of parameters, expensive computational cost, and a huge amount of memory footprint limit their practice in the real world (Ahn et al., 2018).

Storage-efficient super-resolution methods (Kim et al., 2016a; Li et al., 2019; Tai et al., 2017a;b; Li et al., 2020) reuse the parameters recursively to reduce the parameter number or use predefined filters. Computational-efficient methods utilize group convolutions (Hui et al., 2018; Liu et al., 2021), depth-wise separable convolutions (Timofte et al., 2017), hierarchical sub-blocks (Hui et al., 2019), knowledge distillation (He et al., 2020; Gao et al., 2018), re-parameterization technique (Bhardwaj et al., 2021), or automated neural architecture search (Chu et al., 2020) to design lightweight models. However, these lightweight models are not friendly to hardware, leading to slow inference speed in the application. FSRCNN (Dong et al., 2016b) and SRCNN (Dong et al., 2016a) have hardware-friendly network architectures, but current SR methods can significantly outperform them. In comparison, our method trains a hardware-friendly plain network and achieves high performance at the same time.

2.2. Plain Model Training

There are some researches for training a high-performance plain model. An initialization method (Xiao et al., 2018) proposed for classification can train an extremely deep plain network. However, the performance is no better than Vgg16 (Simonyan & Zisserman, 2015) on CIFAR-10 (Krizhevsky et al.). A following work (Oyedotun et al., 2020) combined several techniques to reach 74.6% accuracy on ImageNet (Deng et al., 2009). Recently RepVGG (Ding et al., 2021) significantly improved plain model performance with a re-parameterization technique on classification and segmentation, but not super-resolution. In this paper, we ex-

plore the efficient training method for plain models in SISR.

2.3. Knowledge Distillation

As a well-known method in deep learning, knowledge distillation was proposed by Hinton (Hinton et al., 2015) in 2015. Knowledge distillation uses a trained teacher model to guide the training of a student model in order to enhance the performance in tasks. Researchers have proposed various methods to achieve this goal by transferring the knowledge on logits (Hinton et al., 2015), feature maps (Romero et al., 2015), attention (Zagoruyko & Komodakis, 2017), metrics (Yu et al., 2019), and the relation between samples (Park et al., 2019). While there are hundreds of knowledge distillation methods on the classification task, few can be applied in the super-resolution. SRKD (Gao et al., 2018) is the preliminary work applying knowledge distillation in super-resolution. Nevertheless, it requires the teacher and the student to use a specified network structure. PISR (Lee et al., 2020) pre-trains the teacher with an auto-encoder on the high-resolution image, makes a copy of the trained teacher as the student, then uses adopts feature distillation to improve the student's performance. However, PISR requires the teacher and the student to have the same structure.

3. Method

In this section, we describe the details of our proposed framework. The framework consists of two stages: (1) We derive the winning initialization for the plain student model from a trained teacher model as shown in Figure 2. (2) Then we fine-tune the student using feature distillation coupled with the winning initialization.

3.1. Teacher Structure

The backbone of the teacher follows the design of EDSR (Lim et al., 2017), which is a stack of ResNet-like blocks. We directly apply the pixel-shuffle (Shi et al., 2016) operation in the up-sampling part to reduce the maximum memory footprint. The backbone of the student model is initialized with the proposed method described in the following subsections. In contrast to the backbone, the up-sampling part of the student model is initialized with weights from the up-sampling part in the teacher. More detailed teacher structure can be found in the appendix.

3.2. Convert the Teacher Backbone to an Equivalent Plain Network

In order to convert the teacher backbone to an equivalent plain network, we need to find a way to merge convolutions.

Notations. Denote the kernel and bias of convolution C as $K^{(C)}$ and $b^{(C)}$, respectively. We use $*$ to denote the convolution operation. When $*$ is used between a kernel and a feature map, it is defined as applying a convolution with this kernel on the feature map. So we can express applying a convolution C on a feature map F as $C * F$ or $K^{(C)} * F + b^{(C)}$. We use $\text{Conv}(C_{out}, C_{in}, k)$ to denote a convolution with C_{in} input channels, C_{out} output channels, kernel size $k \times k$, and stride $\frac{k}{2}$. The size of kernel and bias of this convolution are $C_{out} \times C_{in} \times k \times k$ and C_{out}, respectively. The kernel size of all convolutions in the paper is odd.

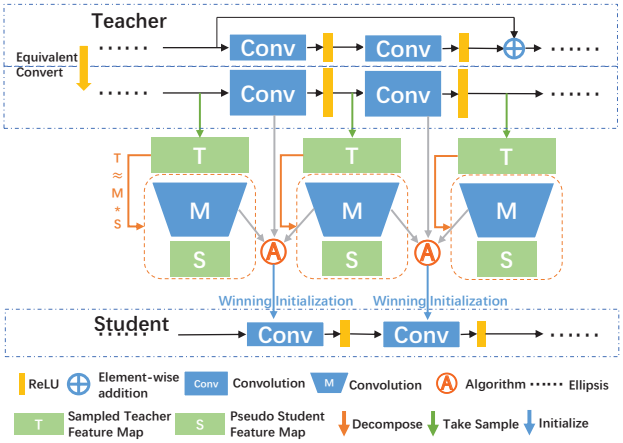

Figure 2. **Overview of the initialization stage in our framework.** We convert the teacher model to an equivalent plain network. Then we use the sampled teacher features to generate the pseudo student features, and derive the student's convolution kernel by our algorithm. (Best viewed in color.)

We define the $\text{pad}_h(X)$ operation on a $t \times t$ matrix X (when X is a scalar, we regard it as $t = 1$) as applying zero-padding with $\frac{h-t}{2}$ zeros on all edges.

We define the **identity convolution** $\text{ID} := \text{Conv(C,C,1)}$ as $K^{(\text{ID})}_{i,i,:,:} = 1, K^{(\text{ID})}_{others} = 0, \forall 1 \le i \le C$ and $b^{(\text{ID})} = 0$, which has the same output as input.

3.2.1. Merge Parallel Convolutions

Let $A := \text{Conv}(C_2, C_1, k_a)$, $B := \text{Conv}(C_4, C_3, k_b)$. We will merge A and B into a convolution C where the output of C is the concatenation of the output of A and B.

When the convolutions A and B have different inputs, we define $C := \text{Conv}(C_2 + C_4, C_1 + C_3, \max\{k_a, k_b\})$ as:

$$K^{(C)}_{i,j,:,:} = \begin{cases} \text{pad}_{k_C}(K^{(A)}_{i,j,:,:}) & \text{if } i \le C_2, j \le C_1 \\ \text{pad}_{k_C}(K^{(B)}_{i-C_2,j-C_1,:,:}) & \text{if } C_2 < i, C_1 < j \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$b^{(C)}$ is the concatenation of $b^{(A)}$ and $b^{(B)}$.

When the convolution A and B share the same input ($C_1 =$

C_3), we construct $C := \text{Conv}(C_2 + C_4, C_1, \max\{k_a, k_b\})$ as:

$$K^{(C)}_{i,j,:,:} = \begin{cases} \text{pad}_{k_C}(K^{(A)}_{i,j,:,:}) & \text{if } i \le C_2 \\ \text{pad}_{k_C}(K^{(B)}_{i-C_2,j,:,:}) & \text{otherwise,} \end{cases} \quad (2)$$

and the $b^{(C)}$ is the same as above.

3.2.2. Merge 1×1 convolution with the following convolution

We define the bias-remove operation to ensure the correctness of merging convolutions as below:

Let $T := \text{Conv}(C_2, C_1, k_t)$, and denote the input feature map as $F \in \mathbb{R}^{N\times C_1\times H\times W}$. We use the following operation to remove $b^{(T)}$: Append a constant channel filled with 1 to the front of F yielding $F' \in \mathbb{R}^{N\times(C_1+1)\times H\times W}$ with

$$F'_{:,i,:,:} = \begin{cases} 1 & \text{if } i = 1 \\ F_{:,i-1,:,:} & \text{otherwise.} \end{cases} \quad (3)$$

Construct $T' := \text{Conv}(C_2, C_1 + 1, k_t)$ with kernel

$$K^{(T')}_{:,i,u,v} = \begin{cases} K^{(T)}_{:,i,u,v} & \text{if } 1 < i \\ b^{(K)} & \text{if } i = 1, u = v = \lceil \frac{k_t}{2} \rceil \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

and no bias. It is easy to see $K^{(T)} * F + b = K^{(T')} * F'$. We denote these operations as $F' = \text{one\_pad}(F)$, and $T' = \text{bias\_rm}(T)$, respectively.

Let $A := \text{Conv}(C_2, C_1, 1)$, the successive convolution $B := \text{Conv}(C_3, C_2, k_b)$. Denote the input feature map to A as F.

Lemma 3.1. *We can construct a convolution $C' := Conv(C_3, C_1 + 1, k_b)$ with zero-bias as*

$$K^{(C')}_{i,k,:,:} = \begin{cases} pad_{k_b}(b^{(B)}_i) + \sum_{j=1}^{C_2} K^{(A')}_{j,1,1,1} K^{(B)}_{i,j,:,:} & \text{if } k = 1 \\ \sum_{j=1}^{C_2} K^{(A')}_{j,k,1,1} K^{(B)}_{i,j,:,:} & \text{otherwise,} \end{cases}$$

so that $B * (A * F) = C' * F'$.

By Lemma 3.1, we can merge 1×1 convolution with the following convolution. However, we must use a one_pad operation before feeding feature maps into each convolution in the plain model. Luckily, this will not increase computation significantly.

3.2.3. Convert the teacher to equivalent plain model

In this subsection, we prove that networks with only non-intersecting skip-connections and odd convolution kernel size can be converted to plain networks. [1]

[1]Proofs for propositions in this section can be found in the appendix.

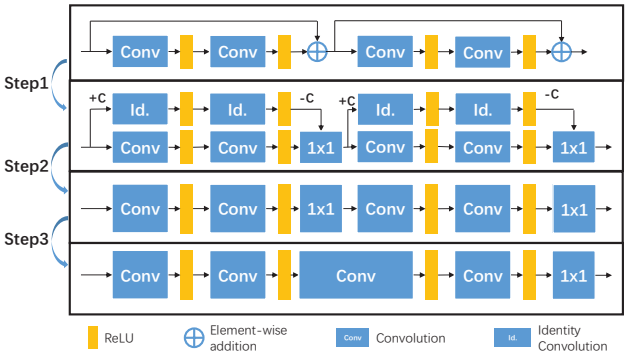

Figure 3. Illustration of converting general models to plain networks. ReLU activation functions in the figure can be replaced with any activation function that preserved the non-negative value. C is the large constant mentioned in Proposition 3.6. Step1: Convert the skip connection into 1×1 convolutions. Step2: Merge the convolutions with different kernel sizes. Step3: Merge neighboring 1×1 convolution and following convolution.

Definition 3.2. *Plain model* is a stack of convolution and ReLU activation.

Definition 3.3. Models with an equivalent plain model are *convertible models*.

Proposition 3.4. *Plain models are convertible model.*

Proposition 3.5. *Stacks of Convertible Models are still a Convertible Model.*

Proposition 3.6. *Let P be a convolution model with non-intersecting skip-connections. Each element of the input feature map to P is limited in $[-C, C](C \in \mathbb{R}^+)$, then P is a convertible model.*

Proof. We use C_1 to denote the input channel of P. First, replace the add operation at the end of the skip connection with an equivalent 1×1 convolution $T = \text{Conv}(C_1, 2C_1, 1)$ with $K^{(T)}_{i,i,:,:} = 1, K^{(T)}_{i,i+C_1,:,:} = 1, K^{(T)}_{others} = 0, b^{(T)} = 0$. Next, substitute the skip connection with a series of identity 1×1 convolutions and add a constant C to the bias of the first convolution, ensuring the feature map in skip-connections to be non-negative. Then add $-C$ to the bias of K to preserve the output. Since the feature map in the skip connection branch is non-negative, we can insert the ReLU activation into the branch. In the end, merge the identity convolution and the corresponding convolution. □

With the above propositions, our EDSR-style teacher model is a *convertible model* and can be converted.

3.3. Calculate the Winning Initialization for the Student

We find the winning initialization of the student from the equivalent plain teacher model in two steps. The first step

is to determine pseudo feature maps for the student. The second step is to calculate the weights for the student with these feature maps.

For simplicity, we regard all activation layers in the teacher and the student as identity layers. Commonly, this hypothesis would introduce some disturbance to the results. However, extensive experiments show that the winning initialization is still useful for the student network.

Denote the post-activation feature maps after each convolution in the backbone teacher model as $F_0^{(T)}, \cdots, F_n^{(T)}$ and $F_0^{(S)}, \cdots, F_n^{(S)}$ for the student (where n is the number of layers in the teacher), where $F_0^{(T)}$ and $F_0^{(S)}$ is the uniformly random sampled input data. Denote the convolution of the teacher model by $C_1^{(T)}, \cdots, C_n^{(T)}$. Similarly, $C_1^{(S)}, \cdots, C_n^{(S)}$ are for the student's convolution layers. Furthermore, n 1×1 convolutions mapping the student features to the teacher features are denoted as $M_0, \cdots, M_n$. M_0 is set to identity convolution since $F_0^{(S)}$ and $F_0^{(T)}$ are the same.

Clearly, we have

$$C_i^{(T)} * F_{i-1}^{(T)} = F_i^{(T)}, \tag{5}$$

$$C_i^{(S)} * F_{i-1}^{(S)} = F_i^{(S)}. \tag{6}$$

We determine the pseudo feature map $F_i^{(S)}$ at the ith ($1 \leq i \leq n$) layer via

$$F_i^{(T)} \approx M^{(i)} * F_i^{(S)}, \tag{7}$$

using the decomposition Algorithm 1.

In Algorithm 1, the coefficient β is introduced to avoid negative values in the pseudo student features, since all post-activation features from the plain student model are non-negative due to ReLU activation.

Along with equations 5, 6, 7 and convolution merge technique in subsection 3.2, we have

$$C_i^{(T)} * F_{i-1}^{(T)} \approx M^{(i)} * F_i^{(S)} \tag{8}$$

$$C_i^{(T)} * (M_{i-1} * F_{i-1}^{(S)}) \approx M^{(i)} * (C_i^{(S)} * F_{i-1}^{(S)}) \tag{9}$$

Merge $C_i^{(T)}$ and M_{i-1} into H using the method in subsection 3.2. Let $F_{i-1}^{(S)\prime} = \text{one\_pad}(F_{i-1}^{(S)})$, $H' = \text{bias\_rm}(H)$, and $C_i^{(S)\prime} = \text{bias\_rm}(C_i^{(S)})$. Then express above equation as

$$H' * F_{i-1}^{(S)\prime} \approx M^{(i)} * (C_i^{(S)\prime} * F_{i-1}^{(S)\prime}). \tag{10}$$

Subtract H' from the bias of $M^{(i)}$ in the first channel, then we can find an approximation for $C_i^{(S)\prime}$ by solving

$$H'_{:,:,u,v} = {M^{(i)}}_{:,:,1,1} {C_i^{(S)\prime}}_{:,:,u,v} \tag{11}$$

Algorithm 1 Determination of M and Pseudo Student Features

Input: Teacher feature map at ith layer $F_i^{(T)}$, the channel width r_i of ith layer in student.
Reshape $F_i^{(T)}$ from (n, c_i, h, w) to $(c_i, n \times h \times w)$;
Calculate the mean value b for each row in $F_i^{(T)}$;
$U, \Sigma, V^T = \text{SVD}(F_i^{(T)} - b)$;
$F_i^{(S)} = \Sigma_{:r_i,:r_i}(V^T)_{:r_i,:}$;
$K = U_{:,:r_i}$;
Apply mean-shift on the pseudo feature map to prevent negative values.
for $k = 1$ **to** r_i **do**
  $\gamma = -\beta$ mean$({F_i^{(S)}}_{k,j}$ where ${F_i^{(S)}}_{k,j} < 0)$;
  ${F_i^{(S)}}_{k,:} = {F_i^{(S)}}_{k,:} + \gamma$;
  for $j = 1$ **to** C_i **do**
    $b_j = b_j - \gamma K_{j,k}$;
  end for
end for
Reshape $F_i^{(S)}$ from $(r_i, N \times H \times W)$ to (N, r_i, H, W);
Reshape K from (C_i, r_i) to $(C_i, r_i, 1, 1)$;
Output: $M^{(i)}$ with K as the kernel and b as the bias, $F_i^{(S)}$.

using least square algorithm.

So far, we have demonstrated the winning initialization algorithm for the student. With this initialization, the plain student's performance can be improved by a large margin with direct training compared with random initialization. (from 25.31dB to 25.97dB PSNR on Urban100 (Huang et al., 2015)). We propose the initialization-aware feature distillation to further boost student performance by make use of $M^{(i)}$ (from 25.97dB to 25.99dB PSNR on Urban100 (Huang et al., 2015)).

3.4. Initialization-Aware Feature Distillation

As most feature distillation methods, we propose the following external loss to encourage the student to learn the knowledge from the teacher:

$$\mathcal{L}_{distill} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}(F_i^{(T)}, M^{(i)} * F_i^{(S)}), \tag{12}$$

where MSE refers to the mean square error, and $M^{(i)}$ are 1×1 convolutions and initialized with weights from Algorithm 1.

Same as existing works, the reconstruction loss is

$$\mathcal{L}_{task} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} |Y_{ij} - Y_{ij}^S|, \tag{13}$$

where H, W is the picture size, Y_{ij} is the HR image and Y_{ij}^S is the SR image from student network.

Learning from the teacher is different from learning from the ground truth. So we apply an exponential decay to prompt the student to learn from the teacher at the start of the training and learn from ground-truth in the end. Overall, we use the following loss to train the student network:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda \epsilon^{\alpha} \mathcal{L}_{distill}, \tag{14}$$

where $\epsilon = 10^{-5}$, $\alpha = \frac{\#current\ iteration}{\#total\ iteration}$, and λ is the coefficient for balancing two parts.

3.5. Dynamic Student Filter Width

Inspired by VGG (Simonyan & Zisserman, 2015), the computation-efficient design of the plain model need to be thin in the early layers and wide in the later layers. One way to determine suitable width for each layer is using automatic neural network search (NAS). However, NAS usually requires a large amount of computation power and time. Our approach efficiently finds the suitable width by calculating the smallest student layer width that approximates the teacher feature maps within limited error.

Denote the input and the results of the algorithm 1 by $F_i^{(T)}, r$ and $M^{(i)}, F_i^{(S)}$, where $F_i^{(T)}$ is the feature map of the teacher model at the ith layer, r is the target filter width at the ith layer of the student model, $F_i^{(S)}$ is the calculated student feature map, and $M^{(i)}$ is the calculated mapping convolution from student to teacher at the ith layer. Then we use binary search to find suitable r that satisfies

$$\frac{\|M^{(i)} * F_i^{(S)} - F_i^{(T)}\|_2}{\|F_i^{(T)}\|_2} \leq \theta, \tag{15}$$

where θ is the error limits of the algorithm. In this way, we calculate the width for each layer. Experiments in the following section show that this can achieve a better speed-PSNR trade-off compared to the equal-width model.

The smaller the θ, the higher the computation and the higher the accuracy the student model has. The inference time of $PlainX_{\theta=0.14}$ is close to that of many lightweight models (IDN (Hui et al., 2018), RFDN (Liu et al., 2020), LatticeNet (Luo et al., 2020), and IMDN (Hui et al., 2019)). $PlainX_{\theta=0.05}$ has better performance and smaller memory footprint than other lightweight models. Therefore we used 0.05 and 0.14 as the value for θ in our experiments.

4. Experiments

4.1. Experimental Details

Datasets. We use the DIV2K (Agustsson & Timofte, 2017) dataset to train our network, which includes 800 pairs of LR and HR images and is widely used in image restoration tasks. We evaluate our framework on standard benchmarks, including Set5 (Bevilacqua et al., 2012), Set14 (Zeyde et al., 2010), B100 (Martin et al., 2001), and Urban100 (Huang et al., 2015).

Implementation Details. The backbone of the teacher network is the same as EDSR (Lim et al., 2017), while the up-sampling part is directly using pixel-shuffle (Shi et al., 2016) operation. There are two versions (S, M) of teacher networks and three versions (S, M, X) of student networks of different sizes. Only Plain-X uses the dynamic filter width in subsection 3.5, and others use equal-width structure. The teacher of Plain-S is Teacher-S, and the teacher of Plain-M and Plain-X is Teacher-M. The details and more experiments are available in the appendix.

Training. DIV2K is used as the training dataset. We randomly crop HR patches of size 192×192 from the HR images. LR patches are cropped from the corresponding LR images by the scale factor. Standard data augmentation (i.e. random rotation and horizontal flipping) are used as with existing works (Hui et al., 2019; Lee et al., 2020; Lim et al., 2017; Hui et al., 2018; Luo et al., 2020; Muqeet et al., 2020). The teacher network is directly trained with random initialization. For training plain student network with our framework, we use $\beta = 3$ in Algorithm 1 which is enough for the algorithm, check appendix for the proof. And we use $C = 1000$ in Figure 3, $\lambda = 0.3$ in equation 14. All super-resolution models in this paper is trained with a batch size of 16 and 10^6 iterations in total. We use the Adam (Kingma & Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as optimizer. For learning rate, we use a one-cycle learning rate scheduler (Smith & Topin, 2019) from PyTorch (Paszke et al., 2019) with maximum learning rate 2×10^{-4}. All experiments using our framework are repeated 4 times with global seeds 233, 234, 235, and 236.

Evaluation. We evaluate the performance of the super-resolved images using peak signal to-noise ratio (PSNR) and structure similarity index (SSIM) (Wang et al., 2004). As with existing works (Ahn et al., 2018; Hui et al., 2019; Dong et al., 2016b; Mei et al., 2020; Niu et al., 2020; Hui et al., 2018) we calculate the values on the luminance channel. No self-ensemble is used during the evaluation. We provide model size, FLOPs (a.k.a. Mult-Adds), inference time, and maximum memory footprint comparisons.

4.2. Ablation Analysis

In this subsection, We present an ablation analysis on each component of our framework. The structure of the teacher network is Teacher-M and the structure of student network is Plain-M. We report quantitative results in terms of the average PSNR on standard benchmarks with the scale factor of 4. In Table 1, we show the average PSNR of student

Table 1. Average PSNR of student networks trained with variants of our framework on standard benchmarks.

Init	FD	FDInit	Set5	Set14	B100	Urban100
×	×	×	31.61	28.22	27.31	25.30
√	×	×	32.09	28.54	27.53	25.97
×	√	×	31.83	28.39	27.44	25.67
×	√	√	31.82	28.40	27.44	25.67
√	√	×	32.08	28.56	27.55	25.99
√	√	√	32.10	28.57	27.55	25.99

trained with different configurations of our framework. The first row shows accuracy of directly training the plain student network.

Winning Initialization (Init). To demonstrate the effect of winning initialization from the teacher. We replace the default initialization of convolutions in the student model with our winning initialization. From the first and the second row, we can see that our initialization significantly improves the plain model's performance. The PSNR of baseline on Set5 increased by $0.5dB$ with only initialization changes and no extra components or parameters.

Feature Distillation (FD). We can see from the first and third row that the distillation in section 3.4 can improve the student performance as well, but is less effective than the initialization. The second and the last row show that these two optimizations are not in conflict, and the winning initialization is more important.

Initialization to Feature Distillation (FDInit). To evaluate the effect of FDInit in equation 12, we compare the training result with the case when using a random learn-able convolution M, which is equivalent to a standard feature distillation pipeline. So we compare M initialization from Algorithm 1 and from random initialization to show the advantages of further coupling of Algorithm 1 and subsequent distillation. As the result on the third and the fourth row, FDInit improves the distillation performance. The fifth and last row show that FDInit also works when winning initialization is adopted. However, the improvements from FDInit is much less than other techniques.

4.3. Ablation of Student Mean-shift

In this subsection, we further analysis the mean-shift in Algorithm 1. The experiment setting is the same as above.

Mean-shift on the Student Feature (MST). MST improves the performance on Set5 by $0.06dB$ when only the winning initialization is applied by comparing the first row and the second row in Table 2. The third and fourth row shows that MST improves the model performance on all benchmarks when feature distillation is adopted but contributes less. The second, the third, and the fourth row show that MST may have a similar effect as the distillation.

Table 2. Quantitative results of whether the student network uses mean-shift in Algorithm 1 or not on standard benchmarks.

Init	FD+FDInit	MST	Set5	Set14	B100	Urban100
√	×	×	32.030	28.519	27.521	25.914
√	×	√	32.090	28.543	27.534	25.969
√	√	×	32.095	28.561	27.546	25.963
√	√	√	32.104	28.566	27.549	25.990

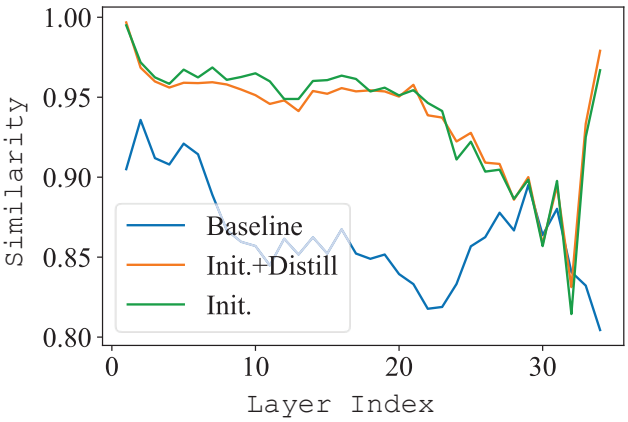

Figure 4. The similarity between the teacher model and the student model. Greater values represent more similar.

4.4. The Similarity Between Student and Teacher

For investigating the knowledge transferred by winning initialization from the teacher to the student, we prepare three types of the student model: (1) A directly trained plain model. (2) A plain model trained with the winning initialization. (3) A plain model trained with the complete training framework. Comparing model (1) and model (2) in Figure 4, we observe that the winning initialization transfer the most knowledge from the teacher to the student. Comparing model (2) and model (3), we find that when both the initialization and the feature distillation are adopted, the student model is more similar to the teacher model in deeper layers.

4.5. Training Curve of the Wining Initialization

In Figure 5, the similarity is measured using centered kernel alignment (Kornblith et al., 2019) on the B100. We can observe that the plain model with winning initialization converges much faster than the common plain model, which demonstrates the effectiveness of our framework.

4.6. Comparison to Existing Training Methods

4.6.1. Comparison to Other SR Distillation Methods

In this section, we compare the distillation result with existing distillation methods in super-resolution. Plain-S and Plain-M are used as student network structures in distilla-

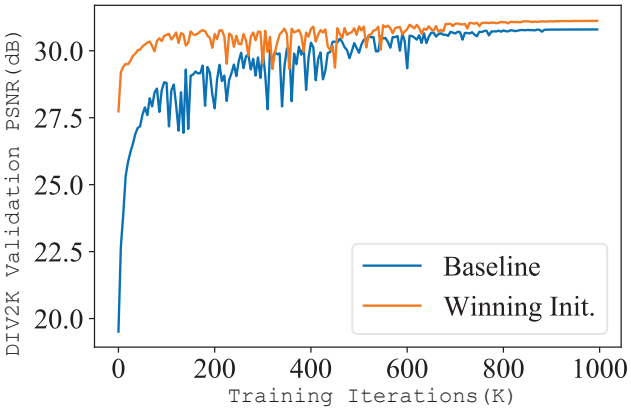

Figure 5. The training curve for a plain model with and without the winning initialization evaluated on DIV2K (Agustsson & Timofte, 2017) validation set. Feature distillation is not adopted.

Table 3. Quantitative results of the student network trained with SRKD (Gao et al., 2018), FAKD (He et al., 2020), and PISR (Lee et al., 2020). The average PSNR for ×4 scale factors on standard benchmarks are reported. Best and second best results are highlighted and underlined.

Depth	Method	Set5	Set14	B100	Urban100
19	Baseline	31.593	28.237	27.315	25.320
	SRKD	30.650	27.635	27.022	24.833
	FAKD	31.609	28.246	27.326	25.342
	PISR	31.219	28.043	27.182	25.057
	Ours	**31.884**	**28.432**	**27.464**	**25.698**
35	Baseline	31.607	28.221	27.314	25.301
	SRKD	31.405	28.122	27.243	25.161
	FAKD	31.587	28.222	27.308	25.303
	PISR	7.615	7.210	8.023	7.169
	Ours	**32.104**	**28.566**	**27.549**	**25.990**

tion. From Table 3, we observe that PISR failed in Plain-M due to gradient explosion problem in the pre-training stage. Moreover, PISR performs poorly though it does not corrupt in Plain-S. In all four datasets, our method far outperforms the existing methods in the plain network distillation.

4.6.2. Comparison to RepVGG

RepVGG (Ding et al., 2021) trains the plain network with the re-parameterization technique and is effective in both classification and segmentation tasks. Here we apply the RepVGG technique in super-resolution by replacing each convolution in the student network with the RepBlock structure in RepVGG and directly train the student network. As the common-sense in super-resolution, batch normalization will harm the performance (Lim et al., 2017). Thus, we remove the batch normalization in the RepVGG structure and name it RepVGG-bn-free. As shown in Table 4, RepVGG-bn-free vastly outperforms the RepVGG in super-resolution, but RepVGG-bn-free still can not train a plain model well.

Table 4. Quantitative results of the student network using RepVGG (Ding et al., 2021) training method. The average PSNR for ×4 scale factors on standard benchmarks are reported. Best and second best results are highlighted and underlined. The baseline represents the same network architecture but is directly trained without any proposed methods.

Student	Method	Set5	Set14	B100	Urban100
Plain-S	Baseline	31.59	28.24	27.32	25.32
	RepVGG	30.44	27.52	26.85	24.52
	RepVGG-bn-free	31.62	28.26	27.33	25.35
	Ours	**31.88**	**28.43**	**27.46**	**25.70**
Plain-M	Baseline	31.61	28.22	27.31	25.30
	RepVGG	30.60	27.61	26.92	24.55
	RepVGG-bn-free	31.62	28.26	27.33	25.35
	Ours	**32.10**	**28.57**	**27.55**	**25.99**

4.7. Comparison With Baseline Methods

4.7.1. Quantitative Comparison on Datasets

In Table 5, we compare the performance of our student model with the state of the art on standard benchmarks including Set5, Set14, BSD100 and Urban100, particularly for efficient super-resolution methods (Dong et al., 2016a; Ahn et al., 2018; Hui et al., 2018; 2019; Dong et al., 2016b; Lim et al., 2017; Luo et al., 2020; Liu et al., 2020; Li et al., 2020). [2]

For a quantitative comparison, we report the average PSNR and SSIM for upsampling factors of 2, 3, and 4, on Set5 (Bevilacqua et al., 2012), Set14 (Zeyde et al., 2010), B100 (Martin et al., 2001), and Urban100 (Huang et al., 2015). We also report the number of model parameters, operations (FLOPs), inference time on NVIDIA GTX 1080Ti, maximum GPU memory footprint in PyTorch, required to reconstruct a HR image of size 1920 × 1080.

As we can observe from this table that Plain-M trained with the proposed framework only requires memories sightly more than FSRCNN (Dong et al., 2016b) while outperforming it by a large margin in all cases. Furthermore, compared to models with comparable performance and inference speed, Plain-$X_{\theta=0.14}$ and Plain-$X_{\theta=0.05}$ only have less than half of the maximum memory footprint. For example, IMDN (Hui et al., 2019) requires 476MB memories and 38ms runtime to achieve the average PSNR of 26.04dB PSNR on Urban100 (Huang et al., 2015) for a factor of 4. In contrast, Plain-$X_{\theta=0.14}$ trained with the proposed framework requires 41ms runtime and only 168MB memories, achieving the average PSNR of 26.05dB. For another example, Plain-$X_{\theta=0.05}$ has faster runtime compared to CARN (Ahn et al., 2018) and EDSR (Lim et al., 2017)

[2] LatticeNet refers to the results given in https://github.com/ymff0592/super-resolution/, which is re-trained on DIV2K dataset and evaluated without self-ensemble.

Table 5. Average PSNR/SSIM for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, BSD100, and Urban100. Best and second best results are highlighted and underlined.

Scale	Method	Param.	FLOPs	Runtime	Memory	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
×2	Bicubic	*	*	*	*	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403
	FSRCNN	24K	34G	13ms	**368MB**	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	29.88 / 0.9020
	Plain-M	1,219K	636G	158ms	420MB	37.99 / 0.9606	33.64 / 0.9184	32.19 / 0.8999	32.10 / 0.9284
	Plain-X$_{\theta=0.05}$	2,224K	1,180G	253ms	718MB	**38.10 / 0.9609**	**33.71 / 0.9187**	**32.24 / 0.9004**	**32.31 / 0.9302**
	SRCNN	68K	144G	60ms	807MB	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946
	IDN	577K	393G	138ms	916MB	37.83 / 0.9600	33.30 / 0.9148	32.08 / 0.8985	31.27 / 0.9196
	EDSR	1,370K	713G	199ms	1,306MB	37.99 / 0.9604	33.57 / 0.9175	32.16 / 0.8994	31.98 / 0.9272
	LatticeNet	765K	384G	178ms	1,710MB	38.06 / 0.9607	33.70 / 0.9187	32.20 / 0.8999	32.25 / 0.9288
	RFDN	534K	279G	166ms	1,731MB	38.05 / 0.9606	33.68 / 0.9184	32.16 / 0.8994	32.12 / 0.9278
	IMDN	694K	359G	150ms	1,813MB	38.00 / 0.9605	33.63 / 0.9177	32.19 / 0.8996	32.17 / 0.9283
	CARN	1,592K	503G	199ms	3,210MB	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.92 / 0.9256
	LARAR	548K	384G	211ms	4,098MB	38.01 / 0.9605	33.62 / 0.9183	32.19 / 0.8999	32.10 / 0.9283
×3	Bicubic	*	*	*	*	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349
	FSRCNN	24K	30G	6ms	**179MB**	33.18 / 0.9140	29.37 / 0.8240	28.53 / 0.7910	26.43 / 0.8080
	Plain-M	1,219K	285G	68ms	202MB	34.34 / 0.9269	30.31 / 0.8417	29.08 / 0.8048	28.10 / 0.8513
	Plain-X$_{\theta=0.05}$	2,379K	561G	116ms	341MB	**34.47 / 0.9278**	**30.37 / 0.8426**	**29.13 / 0.8062**	**28.29 / 0.8552**
	IDN	577K	239G	61ms	435MB	34.11 / 0.9253	29.99 / 0.8354	28.95 / 0.8013	27.42 / 0.8359
	RFDN	541K	125G	72ms	784MB	34.41 / 0.9273	30.34 / 0.8420	29.09 / 0.8050	28.21 / 0.8525
	LatticeNet	765K	172G	79ms	787MB	34.40 / 0.9272	30.32 / 0.8416	29.10 / 0.8049	28.19 / 0.8513
	SRCNN	68K	144G	60ms	807MB	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989
	IMDN	703K	162G	65ms	822MB	34.36 / 0.9270	30.32 / 0.8417	29.09 / 0.8046	28.17 / 0.8519
	EDSR	1,555K	361G	101ms	1,160MB	34.37 / 0.9270	30.28 / 0.8417	29.09 / 0.8052	28.15 / 0.8527
	CARN	1,592K	268G	102ms	1,950MB	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	28.06 / 0.8493
	LARAR	594K	256G	144ms	4,092MB	34.36 / 0.9267	30.34 / 0.8421	29.11 / 0.8054	28.15 / 0.8523
×4	Bicubic	*	*	*	*	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577
	FSRCNN	24K	30G	4ms	**111MB**	30.72 / 0.8660	27.61 / 0.7550	26.98 / 0.7150	24.62 / 0.7280
	Plain-M	1,219K	162G	40ms	161MB	32.10 / 0.8938	28.57 / 0.7805	27.55 / 0.7352	25.99 / 0.7830
	Plain-X$_{\theta=0.14}$	1,259K	167G	41ms	168MB	32.14 / 0.8941	28.59 / 0.7810	27.56 / 0.7356	26.05 / 0.7845
	Plain-X$_{\theta=0.05}$	2,541K	337G	70ms	211MB	**32.21 / 0.8950**	**28.63 / 0.7822**	27.60 / **0.7369**	**26.17 / 0.7883**
	IDN	577K	185G	36ms	269MB	31.82 / 0.8903	28.25 / 0.7730	27.41 / 0.7297	25.41 / 0.7632
	RFDN	550K	72G	43ms	457MB	32.24 / 0.8952	28.61 / 0.7819	27.57 / 0.7360	26.11 / 0.7858
	LatticeNet	777K	98G	47ms	473MB	32.18 / 0.8943	28.61 / 0.7812	27.57 / 0.7355	26.14 / 0.7844
	IMDN	715K	92G	38ms	476MB	32.21 / 0.8948	28.58 / 0.7811	27.56 / 0.7353	26.04 / 0.7838
	SRCNN	68K	143.6G	60ms	807MB	30.48 / 0.8628	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221
	EDSR	1,518K	257G	74ms	1,108MB	32.09 / 0.8938	28.58 / 0.7813	27.57 / 0.7357	26.04 / 0.7849
	CARN	1,592K	205G	76ms	1,554MB	32.13 / 0.8937	28.60 / 0.7806	27.58 / 0.7349	26.07 / 0.7837
	LARAR	659K	211G	122ms	4,090MB	32.15 / 0.8944	28.61 / 0.7818	**27.61** / 0.7366	26.14 / 0.7871

and outperforms them in all the cases. These demonstrate the effectiveness of our approach to training a plain network.

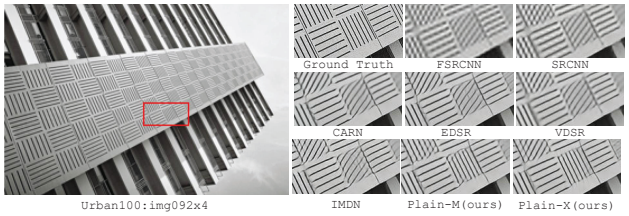

Figure 6. Visual comparisons with other SR methods on Urban100.

4.7.2. Qualitative Comparison on Urban100

Figure 6 shows visual comparisons on the Urban100 dataset. For the "img092" image from Urban100, we can see that most of the compared methods failed to recover direction and details. In contrast, Plain-X correctly recovers the direction and details outperforming others, which qualitatively demonstrates the effectiveness of our framework.

5. Conclusion

This paper present a novel plain network training framework for lightweight and accurate single image super-resolution. We show with experiments that the plain network trained with MemSR requires less than half the memory footprint while having comparable performance and inference speed compared to the state-of-arts in lightweight models. The detailed analysis of each component of our framework demonstrates the effectiveness of our approach.

References

Agustsson, E. and Timofte, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

Ahn, N., Kang, B., and Sohn, K. Fast, accurate, and lightweight super-resolution with cascading residual network. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, volume 11214 of *Lecture Notes in Computer Science*, pp. 256–272. Springer, 2018. doi: 10.1007/978-3-030-01249-6_16. URL https://doi.org/10.1007/978-3-030-01249-6_16.

Bevilacqua, M., Roumy, A., Guillemot, C., and Alberi-Morel, M. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In Bowden, R., Collomosse, J. P., and Mikolajczyk, K. (eds.), *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, pp. 1–10. BMVA Press, 2012. doi: 10.5244/C.26.135. URL https://doi.org/10.5244/C.26.135.

Bhardwaj, K., Milosavljevic, M., Chalfin, A., Suda, N., O'Neil, L., Gope, D., Meng, L., Navarro, R. M., and Loh, D. Collapsible linear blocks for super-efficient super resolution. *CoRR*, abs/2103.09404, 2021. URL https://arxiv.org/abs/2103.09404.

Chu, X., Zhang, B., Ma, H., Xu, R., and Li, Q. Fast, accurate and lightweight super-resolution with neural architecture search. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*, pp. 59–64. IEEE, 2020. doi: 10.1109/ICPR48806.2021.9413080. URL https://doi.org/10.1109/ICPR48806.2021.9413080.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. Repvgg: Making vgg-style convnets great again. *CoRR*, abs/2101.03697, 2021. URL https://arxiv.org/abs/2101.03697.

Dong, C., Loy, C. C., He, K., and Tang, X. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016a. doi: 10.1109/TPAMI.2015.2439281. URL https://doi.org/10.1109/TPAMI.2015.2439281.

Dong, C., Loy, C. C., and Tang, X. Accelerating the super-resolution convolutional neural network. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, volume 9906 of *Lecture Notes in Computer Science*, pp. 391–407. Springer, 2016b. doi: 10.1007/978-3-319-46475-6_25. URL https://doi.org/10.1007/978-3-319-46475-6_25.

Du, Z., Liu, J., Tang, J., and Wu, G. Anchor-based plain net for mobile image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pp. 2494–2502. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPRW53098.2021.00283. URL https://openaccess.thecvf.com/content/CVPR2021W/MAI/html/Du_Anchor-Based_Plain_Net_for_Mobile_Image_Super-Resolution_CVPRW_2021_paper.html.

Gao, Q., Zhao, Y., Li, G., and Tong, T. Image super-resolution using knowledge distillation. In Jawahar, C. V., Li, H., Mori, G., and Schindler, K. (eds.), *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part II*, volume 11362 of *Lecture Notes in Computer Science*, pp. 527–541. Springer, 2018. doi: 10.1007/978-3-030-20890-5_34. URL https://doi.org/10.1007/978-3-030-20890-5_34.

Haris, M., Shakhnarovich, G., and Ukita, N. Deep back-projection networks for super-resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1664–1673. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00179. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Haris_Deep_Back-Projection_Networks_CVPR_2018_paper.html.

He, Z., Dai, T., Lu, J., Jiang, Y., and Xia, S. Fakd: Feature-affinity based knowledge distillation for efficient image super-resolution. In *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*, pp. 518–522. IEEE, 2020. doi: 10.1109/ICIP40778.2020.9190917. URL https://doi.org/10.1109/ICIP40778.2020.9190917.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt*

Lake City, UT, USA, June 18-22, 2018, pp. 7132–7141. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00745. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.243. URL https://doi.org/10.1109/CVPR.2017.243.

Huang, J., Singh, A., and Ahuja, N. Single image super-resolution from transformed self-exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 5197–5206. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7299156. URL https://doi.org/10.1109/CVPR.2015.7299156.

Hui, Z., Wang, X., and Gao, X. Fast and accurate single image super-resolution via information distillation network. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 723–731. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00082. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Hui_Fast_and_Accurate_CVPR_2018_paper.html.

Hui, Z., Gao, X., Yang, Y., and Wang, X. Lightweight image super-resolution with information multi-distillation network. In Amsaleg, L., Huet, B., Larson, M. A., Gravier, G., Hung, H., Ngo, C., and Ooi, W. T. (eds.), *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pp. 2024–2032. ACM, 2019. doi: 10.1145/3343031.3351084. URL https://doi.org/10.1145/3343031.3351084.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

Kim, J., Lee, J. K., and Lee, K. M. Deeply-recursive convolutional network for image super-resolution. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1637–1645. IEEE Computer Society, 2016a. doi: 10.1109/CVPR.2016.181. URL https://doi.org/10.1109/CVPR.2016.181.

Kim, J., Lee, J. K., and Lee, K. M. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1646–1654. IEEE Computer Society, 2016b. doi: 10.1109/CVPR.2016.182. URL https://doi.org/10.1109/CVPR.2016.182.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. Similarity of neural network representations revisited. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., and Shi, W. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 105–114. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.19. URL https://doi.org/10.1109/CVPR.2017.19.

Lee, W., Lee, J., Kim, D., and Ham, B. Learning with privileged information for efficient image super-resolution. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIV*, volume 12369 of *Lecture Notes in Computer Science*, pp. 465–482. Springer, 2020. doi: 10.1007/978-3-030-58586-0_28. URL https://doi.org/10.1007/978-3-030-58586-0_28.

Li, W., Zhou, K., Qi, L., Jiang, N., Lu, J., and Jia, J. LAPAR: linearly-assembled pixel-adaptive regression network for single image super-resolution and beyond. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural*

Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/eaae339c4d89fc102edd9dbdb6a28915-Abstract.html.

Li, Z., Yang, J., Liu, Z., Yang, X., Jeon, G., and Wu, W. Feedback network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 3867–3876. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00399. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Li_Feedback_Network_for_Image_Super-Resolution_CVPR_2019_paper.html.

Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1132–1140. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.151. URL https://doi.org/10.1109/CVPRW.2017.151.

Liu, J., Tang, J., and Wu, G. Residual feature distillation network for lightweight image super-resolution. In Bartoli, A. and Fusiello, A. (eds.), *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12537 of *Lecture Notes in Computer Science*, pp. 41–55. Springer, 2020. doi: 10.1007/978-3-030-67070-2_2. URL https://doi.org/10.1007/978-3-030-67070-2_2.

Liu, X., Li, Y., Fromm, J., Wang, Y., Jiang, Z., Mariakakis, A., and Patel, S. N. Splitsr: An end-to-end approach to super-resolution on mobile devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1):25:1–25:20, 2021. doi: 10.1145/3448104. URL https://doi.org/10.1145/3448104.

Luo, X., Xie, Y., Zhang, Y., Qu, Y., Li, C., and Fu, Y. Latticenet: Towards lightweight image super-resolution with lattice block. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, volume 12367 of *Lecture Notes in Computer Science*, pp. 272–289. Springer, 2020. doi: 10.1007/978-3-030-58542-6_17. URL https://doi.org/10.1007/978-3-030-58542-6_17.

Mao, X., Shen, C., and Yang, Y. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2802–2810, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/0ed9422357395a0d4879191c66f4faa2-Abstract.html.

Martin, D. R., Fowlkes, C. C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, July 7-14, 2001 - Volume 2*, pp. 416–425. IEEE Computer Society, 2001. doi: 10.1109/ICCV.2001.937655. URL https://doi.org/10.1109/ICCV.2001.937655.

Mei, Y., Fan, Y., Zhou, Y., Huang, L., Huang, T. S., and Shi, H. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 5689–5698. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00573. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Mei_Image_Super-Resolution_With_Cross-Scale_Non-Local_Attention_and_Exhaustive_Self-Exemplars_Mining_CVPR_2020_paper.html.

Muqeet, A., Hwang, J., Yang, S., Kang, J. H., Kim, Y., and Bae, S. Multi-attention based ultra lightweight image super-resolution. In Bartoli, A. and Fusiello, A. (eds.), *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12537 of *Lecture Notes in Computer Science*, pp. 103–118. Springer, 2020. doi: 10.1007/978-3-030-67070-2_6. URL https://doi.org/10.1007/978-3-030-67070-2_6.

Niu, B., Wen, W., Ren, W., Zhang, X., Yang, L., Wang, S., Zhang, K., Cao, X., and Shen, H. Single image super-resolution via a holistic attention network. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XII*, volume 12357 of *Lecture Notes in Computer Science*, pp. 191–207. Springer, 2020. doi: 10.1007/978-3-030-58610-2_12. URL https://doi.org/10.1007/978-3-030-58610-2_12.

Oyedotun, O. K., Shabayek, A. E. R., Aouada, D., and Ottersten, B. E. Going deeper with neural networks without skip connections. In *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*, pp. 1756–1760. IEEE, 2020. doi: 10.1109/ICIP40778.2020.9191356. URL https://doi.org/10.1109/ICIP40778.2020.9191356.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 3967–3976. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00409. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Park_Relational_Knowledge_Distillation_CVPR_2019_paper.html.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6550.

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1874–1883. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.207. URL https://doi.org/10.1109/CVPR.2016.207.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

Tai, Y., Yang, J., and Liu, X. Image super-resolution via deep recursive residual network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2790–2798. IEEE Computer Society, 2017a. doi: 10.1109/CVPR.2017.298. URL https://doi.org/10.1109/CVPR.2017.298.

Tai, Y., Yang, J., Liu, X., and Xu, C. Memnet: A persistent memory network for image restoration. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4549–4557. IEEE Computer Society, 2017b. doi: 10.1109/ICCV.2017.486. URL https://doi.org/10.1109/ICCV.2017.486.

Timofte, R., Agustsson, E., Gool, L. V., Yang, M., Zhang, L., Lim, B., Son, S., Kim, H., Nah, S., Lee, K. M., Wang, X., Tian, Y., Yu, K., Zhang, Y., Wu, S., Dong, C., Lin, L., Qiao, Y., Loy, C. C., Bae, W., Yoo, J. J., Han, Y., Ye, J. C., Choi, J., Kim, M., Fan, Y., Yu, J., Han, W., Liu, D., Yu, H., Wang, Z., Shi, H., Wang, X., Huang, T. S., Chen, Y., Zhang, K., Zuo, W., Tang, Z., Luo, L., Li, S., Fu, M., Cao, L., Heng, W., Bui, G., Le, T., Duan, Y., Tao, D., Wang, R., Lin, X., Pang, J., Xu, J., Zhao, Y., Xu, X., Pan, J., Sun, D., Zhang, Y., Song, X., Dai, Y., Qin, X., Huynh, X., Guo, T., Mousavi, H. S., Vu, T. H., Monga, V., Cruz, C., Egiazarian, K. O., Katkovnik, V., Mehta, R., Jain, A. K., Agarwalla, A., Praveen, C. V. S., Zhou, R., Wen, H., Zhu, C., Xia, Z., Wang, Z., and Guo, Q. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1110–1121. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.149. URL https://doi.org/10.1109/CVPRW.2017.149.

Tong, T., Li, G., Liu, X., and Gao, Q. Image super-resolution using dense skip connections. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4809–4817. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.514. URL https://doi.org/10.1109/ICCV.2017.514.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://doi.org/10.1109/TIP.2003.819861.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10, 000-layer vanilla convolutional neural networks. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5389–5398. PMLR, 2018. URL http://proceedings.mlr.press/v80/xiao18a.html.

Xiao, X., Wei, P., Mao, W., and Wang, L. Context-aware multi-view attention networks for emotion cause extraction. In *2019 IEEE International Conference on Intelligence and Security Informatics, ISI 2019, Shenzhen, China, July 1-3, 2019*, pp. 128–133. IEEE, 2019. doi: 10.1109/ISI.2019.8823225. URL https://doi.org/10.1109/ISI.2019.8823225.

Yu, L., Yazici, V. O., Liu, X., van de Weijer, J., Cheng, Y., and Ramisa, A. Learning metrics from teachers: Compact networks for image embedding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 2907–2916. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00302. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Yu_Learning_Metrics_From_Teachers_Compact_Networks_for_Image_Embedding_CVPR_2019_paper.html.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sks9_ajex.

Zeyde, R., Elad, M., and Protter, M. On single image scale-up using sparse-representations. In Boissonnat, J., Chenin, P., Cohen, A., Gout, C., Lyche, T., Mazure, M., and Schumaker, L. L. (eds.), *Curves and Surfaces - 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers*, volume 6920 of *Lecture Notes in Computer Science*, pp. 711–730. Springer, 2010. doi: 10.1007/978-3-642-27413-8_47. URL https://doi.org/10.1007/978-3-642-27413-8_47.

Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., and Fu, Y. Image super-resolution using very deep residual channel attention networks. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pp. 294–310. Springer, 2018a. doi: 10.1007/978-3-030-01234-2_18. URL https://doi.org/10.1007/978-3-030-01234-2_18.

Zhang, Y., Tian, Y., Kong, Y., Zhong, B., and Fu, Y. Residual dense network for image super-resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2472–2481. IEEE Computer Society, 2018b. doi: 10.1109/CVPR.2018.00262. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Residual_Dense_Network_CVPR_2018_paper.html.

Zhang, Y., Li, K., Li, K., Zhong, B., and Fu, Y. Residual non-local attention networks for image restoration. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HkeGhoA5FX.

Zhao, H., Kong, X., He, J., Qiao, Y., and Dong, C. Efficient image super-resolution using pixel attention. In Bartoli, A. and Fusiello, A. (eds.), *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12537 of *Lecture Notes in Computer Science*, pp. 56–72. Springer, 2020. doi: 10.1007/978-3-030-67070-2_3. URL https://doi.org/10.1007/978-3-030-67070-2_3.

A. Proofs

A.1. Proof to Lemma 3.1

Proof. Let $A' = bias\_rm(A), F' = one\_\text{pad}(F)$, and we have $B*(A*F) = K^{(B)}*(K^{(A')}*F')+b^{(B)}$. $\forall i \in [1, C_3]$,

$$
\begin{aligned}
&(K^{(B)} * (K^{(A')} * F') + b^{(B)})_{:,i,:,:} \\
&= b_i^{(B)} + \sum_{j=1}^{C_2} K_{i,j,:,:}^{(B)} * (\sum_{k=1}^{C_1} K_{j,k,1,1}^{(A')} \times F'_{:,k,:,:}) \\
&= b_i^{(B)} + \sum_{k=1}^{C_1+1} \sum_{j=1}^{C_2} K_{j,k,1,1}^{(A')} \times (K_{i,j,:,:}^{(B)} * F'_{:,k,:,:}) \\
&= b_i^{(B)} + \sum_{k=1}^{C_1+1} (\sum_{j=1}^{C_2} K_{j,k,1,1}^{(A')} K_{i,j,:,:}^{(B)}) * F'_{:,k,:,:} \\
&\text{by the definition of } K^{(C')}, \\
&= \sum_{k=1}^{C_1+1} K_{i,k}^{(C')} * F'_{:,k,:,:} (Note\ that\ F'_{:,1,:,:} = 1)
\end{aligned}
$$

□

A.2. Proof to Proposition 3.4

Proof. The plain model itself is the *equivalent plain*. Thus Plain models are convertible models according to the definition. □

A.3. Proof to Proposition 3.5

Proof. We can convert *convertible models* to *plain models* and stack them up as a new *plain model*, which is a convertible model by Proposition 3.4. □

A.4. Proof to the choice of β

β is introduced to avoid negative values for the student feature computed in Algorithm 1. This is to reduce the impact of the assumption *All activation layers in the teacher and the student are identity layers*. We further evaluate the necessary of this design in Table 2. When the distribution of the values of $F_i^{(S)}$ before the mean-shift operation in Algorithm 1 follows a normal distribution with zero mean, $\beta = 3$ ensures that $\Phi(\sqrt{\frac{18}{\pi}}) \approx 99.16\%$ of the values of $F_i^{(S)}$ after the mean-shift operation are positive, thus $\beta = 3$ is enough.

B. More discussion

B.1. Ablation of up-sampling part initialization

In this subsection, we show that the model does not take advantage of the up-sampling part initialization from the teacher model. The up-sampling part is not fixed during the training. Table 10 shows that the initialization of the up-sampling part is not an important factor.

Table 6. The average PSNR of student networks trained with different balance parameters λ on standard benchmarks with ×4 scale.

λ	Set5	Set14	B100	Urban100
0.1	32.104	28.557	27.549	25.983
0.3	32.113	28.568	27.549	25.986
0.5	32.101	28.564	27.551	25.983
1.0	32.110	28.563	27.552	25.983
2.0	32.100	28.558	27.550	25.984

Table 7. The average PSNR of student networks trained with different parameters ϵ on standard benchmarks with ×4 scale.

ϵ	Set5	Set14	B100	Urban100
1e-6	32.086	28.558	27.547	25.980
1e-5	32.120	28.571	27.550	25.999
1e-4	32.113	28.565	27.549	25.991
1e-3	32.072	28.546	27.541	25.955

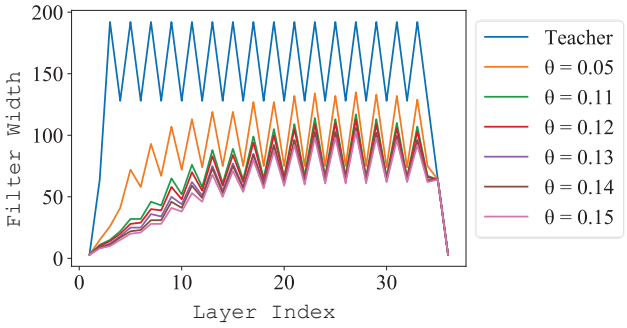

Figure 7. Filter width for each layer of Plain-X with different θ.

B.2. Hyper-parameters

In Table 6 and Table 7, we investigate the influence of the only two hyper-parameters used in our method: (1) the λ for balancing two parts of the loss, and (2) the ϵ in Eq. 14 which is the exponential decay rate of distillation loss. The student network structure is Plain-M.

We validate a series of λ: $0.1, 0.3, 0.5, 1.0$, and 2.0. As shown in Table 6, the performance increases on Set5 as λ gets smaller. But the performance gets worse on Urban100 when λ is too small. And it can be observed that 10^{-5} is the best in Table 7.

B.3. Comparison to other initialization methods

For the winning initialization, we compare it with other convolution neural network initialization techniques. Table B.3

Table 8. **Teacher Structures**.

Name	#features	#resblocks
Teacher-S	64	8
Teacher-M	64	16
Teacher-L	100	16

Table 9. **Student Structures**.

Name	#layers	width
Plain-S	19	64
Plain-M	35	64
Plain-X	35	Dynamic
Plain-L	35	100

Table 10. Average PSNR on standard benchmarks under different initialization without distillation, where ×refers to random initialization.

Backbone	UpSample	Set5	Set14	B100	Urban100
×	×	31.61	28.22	27.31	25.30
×	√	31.61	28.24	27.32	25.33
√	×	32.08	28.54	27.53	25.97
√	√	32.09	28.54	27.53	25.97

shows that these initialization methods perform similarly in SR. None of them can train a plain model well.

C. More details and experiments

C.1. Details of teacher structure and student structure

Tables 8 and 9 show the detail of student structures and teacher structures. '#features' indicate the filter width in the backbone of the teacher. '#resblocks' indicate the number of residual blocks used in the EDSR-like teacher backbone. '#layers' indicate the number of convolutional layers in the student plain model, while 'width' stands for the filter width in the student. Plain-X uses the dynamic filter width decided by hyper-parameter θ mentioned in section 3.5 in the paper. Figure 7 shows the filter width of each layer corresponding to different θ. The 'Teacher' in the figure is the equivalent plain network of Teacher-M.

C.2. More quantitative comparison

Table 12 provides the results for Plain-L and Plain-$X_{\theta=0.14}$ in more cases.

C.3. More qualitative comparison

Figure 8 shows reconstruction examples on B100 (Martin et al., 2001) and Urban100 (Huang et al., 2015) datasets using student networks. We can see that the student networks trained with our framework provide comparable results compared to other light-weighted models, consistent with the quantitative results.

Table 11. Quantitative results of the student network using other initialization methods when training Plain-M. The average PSNR for different scale factors (2×, 3×, and 4×) on standard benchmarks. Best results are highlighted.

Scale	Method	Set5	Set14	B100	Urban100
×4	He. Normal	31.59	28.21	27.31	25.29
	He. Uniform	31.56	28.20	27.30	25.28
	Xavier Normal	31.58	28.22	27.31	25.29
	Xavier Uniform	31.55	28.21	27.30	25.28
	Ours	**32.1**	**28.57**	**27.55**	**26.00**
×3	He. Normal	33.88	29.98	28.83	27.30
	He. Uniform	33.86	29.97	28.83	27.29
	Xavier Normal	33.88	29.99	28.84	27.32
	Xavier Uniform	33.86	29.98	28.83	27.29
	Ours	**34.34**	**30.31**	**29.08**	**28.10**
×2	He. Normal	37.59	33.20	31.92	31.09
	He. Uniform	37.59	33.20	31.92	31.10
	Xavier Normal	37.59	33.19	31.92	31.07
	Xavier Uniform	37.59	33.18	31.91	31.05
	Ours	**37.99**	**33.64**	**32.19**	**32.00**

Table 12. Average PSNR for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, BSD100, and Urban100.

Scale	Method	Param.	FLOPs	Runtime	Memory	Set5	Set14	B100	Urban100
×2	plain-X$_{\theta=0.14}$	935K	496G	134ms	547MB	37.98	33.60	32.17	32.04
	Plain-L	3,005K	1,595G	324ms	641MB	38.08	33.71	32.23	32.28
	SwinIR-S	878K	810G	2,877ms	2910MB	38.14	33.86	32.31	32.76
×3	Plain-X$_{\theta=0.14}$	1106K	261G	65ms	264MB	34.37	30.30	29.09	28.13
	Plain-L	3005K	712G	145ms	305MB	34.42	30.36	29.13	28.24
	SwinIR-S	886K	360G	1,204ms	1,305MB	34.62	30.54	29.20	28.66
×4	Plain-L	3005K	402G	82ms	188MB	32.18	28.62	27.59	26.12
	SwinIR-S	897K	205G	654ms	759MB	32.44	28.77	27.69	26.47

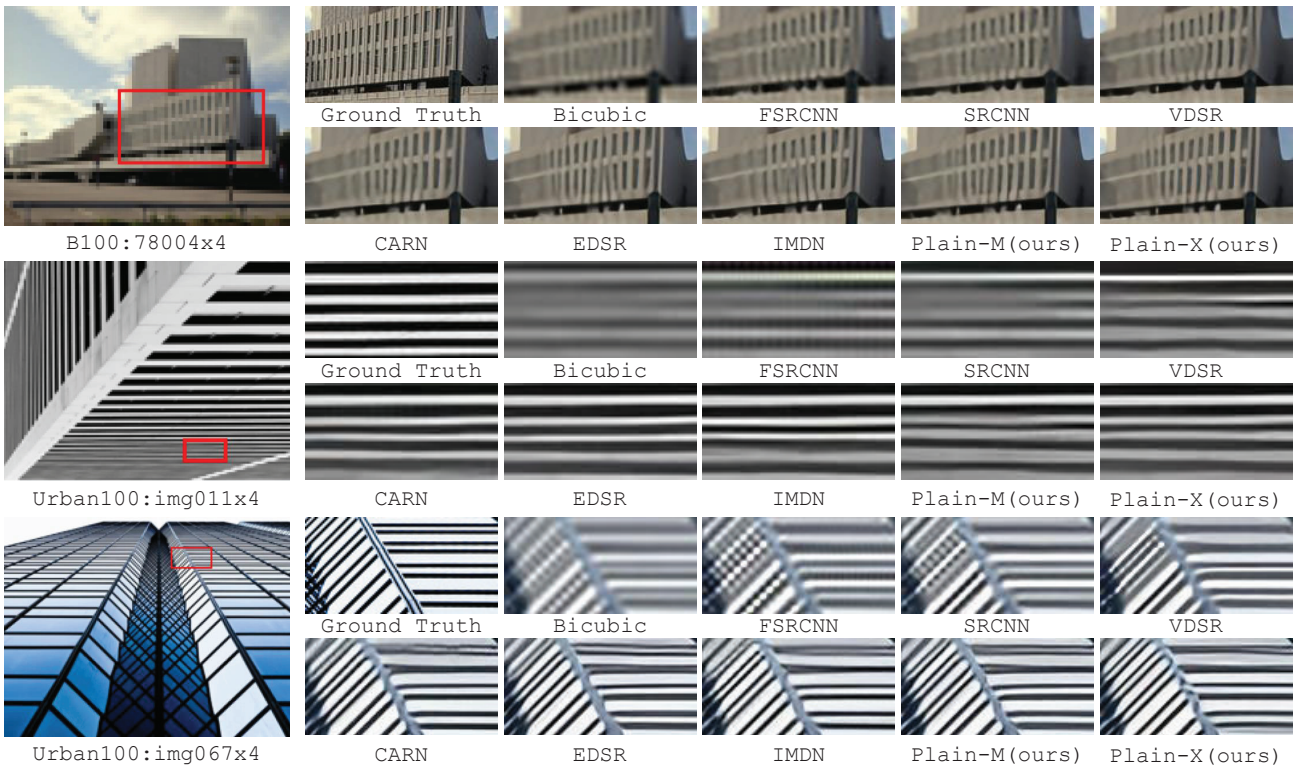

Figure 8. More qualitative comparisons on different datasets.