MFAGAN: A Compression Framework for Memory-Efficient On-Device Super-Resolution GAN

Wenlong Cheng
City University of Hong Kong

Mingbo Zhao
Donghua University

Zhiling Ye
Tencent Computer System Co., Ltd.

Shuhang Gu
The University of Sydney

Abstract

Generative adversarial networks (GANs) have promoted remarkable advances in single-image super-resolution (SR) by recovering photo-realistic images. However, high memory consumption of GAN-based SR (usually generators) causes performance degradation and more energy consumption, hindering the deployment of GAN-based SR into resource-constricted mobile devices. In this paper, we propose a novel compression framework Multi-scale Feature Aggregation Net based GAN (MFAGAN) for reducing the memory access cost of the generator. First, to overcome the memory explosion of dense connections, we utilize a memory-efficient multi-scale feature aggregation net as the generator. Second, for faster and more stable training, our method introduces the PatchGAN discriminator. Third, to balance the student discriminator and the compressed generator, we distill both the generator and the discriminator. Finally, we perform a hardware-aware neural architecture search (NAS) to find a specialized SubGenerator for the target mobile phone. Benefiting from these improvements, the proposed MFAGAN achieves up to 8.3× memory saving and 42.9× computation reduction, with only minor visual quality degradation, compared with ESRGAN. Empirical studies also show ~70 milliseconds latency on Qualcomm Snapdragon 865 chipset.

1. Introduction

Single image super-resolution (SR) is a fundamental low-level vision task, which aims to reconstruct a high-resolution (HR) image from a degraded low-resolution (LR) input. In recent years, convolutional neural networks (CNNs) based approaches [10, 11, 31, 42, 21, 3, 41, 4] have achieved high fidelity in terms of peak signal-to-noise ratio (PSNR) [34]. Nevertheless, these PSNR-oriented methods tend to produce blurry output without enough high-frequency information. More recently, an emerging direction is to resolve the ill-posed SR problem by using generative adversarial networks (GANs) [14]. State-of-the-art GAN-based perceptual-driven models such as SRGAN [23], ESRGAN [36] can generate photorealistic images with more natural textures and sharper edges. With the popularity of mobile devices, there is a growing on-device demand for GAN-based SR applications.

However, the aforementioned GAN-based SR applications are extremely memory intensive and energy overhead, making it impractical to deploy GAN-based SR generators into resource-limited mobile devices. On the one hand, mobile devices are memory-constrained. For instance, a Snapdragon 865 GPU has 2800MB memory bandwidth, while ESRGAN would cost over 4000MB memory consumption to process a 256×256 image. Furthermore, the total memory bandwidth is shared by various on-device applications and the operating system. The peak bandwidth consumed by a single application may only be allocated 20–40% [24] of the total memory bandwidth. The overall performance of GAN-based SR is bound by the limited memory bandwidth and lives under the slanted part of the roofline [28].

On the other hand, mobile devices are energy-constrained.
The energy consumption of GAN-based SR mainly comes from memory access cost and computation. With a 45 nm technology, the 32b coefficients in off-chip DRAM cost 640 pJ, 128 × larger than the consumption of 32b coefficients in on-chip SRAM (3.7 pJ) [15]. The heavy memory cost of GAN-based SR severely hinders the mobile deployment or at least creates performance degradation.

During the past few years, tremendous model compression techniques [15] have been proposed to speed-up the inference and reduce the model parameters and GAN compression is recently a hot topic in this research area [20, 12, 35]. For example, Aguinaldo et al. [1], and Chen et al. [7] first exploited different knowledge distillation modalities for CycleGAN compression. Gong et al. [13], Shu et al. [32], and Chu et al. [8] employed reinforcement learning or co-evolutionary learning-based search to accelerate GAN. However, these methods cannot be directly extended to compress GAN-based SR due to the following reasons: first, the minimax training of GAN is notoriously unstable and prone to collapse. This will greatly result in non-trivial solutions for some complex tasks such as GAN-based SR compression; second, memory efficiency is a critical issue when deploying the memory overhead models on the mobile phone. However, none of above compression methods have consider this point. Therefore, how to develop a memory efficiency and effective generator for GAN-based SR, is an urgent problem.

In this work, in order to develop a hardware-aware on-device GAN-based SR network, we propose a new network, namely, Multi-scale Feature Aggregation Net based GAN (MFAGAN) for memory efficient compression. In detail, we first propose a novel generator architecture by designing the multi-scale feature aggregation modules (MFAMs), which is memory-efficient and of sufficient expressive ability. Besides, we introduce the light-weight PatchGAN discriminator to overcome artifacts and facilitate MFAGAN training. Meriting from the above structures, we propose a two-stage compression method: we first utilize knowledge distillation both for the generator and discriminator. This is to balance the student generator and student discriminator for maintaining the stabilized training of the compressed generator so that the non-trivial solution can be achieved; we then apply NAS channel pruning method to further reduce memory usage. Finally, we utilize the hardware-aware evolutionary search for specializing SubGenerator on the target mobile phone.

The main contributions can be shown as follows: 1) we start from a new and more useful point of view to design a small-size SR method by firstly considering the memory efficiency. This is of great practice in real-world applications; 2) we develop a new two-stage approach to achieve effective compression, where we first distill G/D to achieve stabilized non-trivial solution of the compressed generator and then apply NAS to reduce memory usage; 3) Extensive experiments validate the efficiency of our method. MFAGAN achieves up to 8.3× memory reduction, 42.9× computing efficiency over ESRGAN with only minor loss in PSNR and Learned Perceptual Image Patch Similarity (LPIPS) [34]. Finally, we deploy our MFAGAN on OPPO Find X2 and demonstrate ∼70 milliseconds latency on Snapdragon 865 GPU.

2. Related Work

In this section, we review previous works about GAN-based super-resolution networks, efficient super-resolution networks, and GAN compression techniques, which are the most relevant to our work.

2.1. GAN-based Super-Resolution Networks

Recently, a bunch of SR works paid more attention to visual effects. With the rapid development of perceptual-driven SR algorithms, Generative Adversarial Network (GAN)-based methods often achieved state-of-the-art visual performance. Johnson et al. [20] adopted the perceptual loss to enhance the visual quality while Ledig et al. [23] firstly employed the adversarial loss to generate more realistic images. Besides, Sajjadi et al. [30] explored the texture matching loss to reduce visually unpleasant artifacts. Based on these works, Wang et al. [36] enhanced the SRGAN by employing Residual-in-Residual Dense Block (RRDB) and the relativistic discriminator, won the champion of PIRM2018-SR challenge. Furthermore, the LPIPS metric was introduced to measure the perceptual similarity. Lately, Zhang et al. [40] proposed a novel rank-content loss to optimize the perceptual quality, which achieved state-of-the-art results in perceptual metrics. Despite their performance boost, large GAN generators’ growing complexity conflicts with the demands of mobile deployments, calling for GAN compression techniques. In this paper, we focus on memory-efficient GAN-based SR for mobile applications.

2.2. Efficient Super-Resolution Networks

In recent years, a series of efficient networks with parameters in the range of 10M have been proposed for the efficient SR task [59]. We call these kinds of networks efficient super-resolution networks. They can be approximately divided into two classes: handcrafted architectures and model compression-based methods. A surge of handcrafted architectures have been designed for the efficient SR task, ranging from post-upsampling operators [11], group convolutions [19], residual blocks [42], recursive structures [21], cascaded architectures [3], inverse sub-pixel convolution [31], attention mechanisms [41], to information multi-distillation block (IMDB) [18]. Besides, model compression techniques such as knowledge distillation, channel pruning, and binary quantization have also been used to
speed up the SR networks. Specifically, RFDN [27] applied channel pruning along with residual feature aggregation module to improve the IMDB efficiency, which is the winner solution of the AIM 2020 Challenge on Efficient Super-Resolution [39]. However, these methods aim to maximize PSNR between SR and HR, which tend to generate blurry results without high-frequency details. Efficient super-resolution networks cannot be directly used as the generator of GAN-based SR.

2.3. GAN Compression Techniques

In the past few years, GAN has achieved prevailing success in many generation and translation tasks. However, the growing memory complexity and computation cost of GANs conflict with the demands of mobile deployments. It is hard to apply existing compression techniques, owing to the minimax training of GANs is notoriously unstable and prone to collapse. Several methods exploited knowledge distillation for compressing the image translation models. Aguinaldo et al. [1] first introduced knowledge distillation for unconditional GANs compression. Chen et al. [2] proposed to train an efficient generator by knowledge distillation over the architectures of both the generator and the discriminator. A few works have also attempted to incorporate neural architecture search (NAS) [42] with GANs. Gong et al. [13] utilized reinforcement learning to search for an efficient generator with a fixed discriminator, limiting the algorithm to discover an optimal generator since the balance between these two players needs to be considered. Shu et al. [32] later replaced the reinforcement learning by co-evolutionary pruning to accelerate CycleGAN, which relied on the cycle consistency loss. Chu et al. [8] leveraged an elastic search tactic at both micro and macro space to solve a multi-objective problem for SR. Combining multiple different compression techniques, such as weight sharing, channel pruning, knowledge distillation, and quantization, has been significantly outperformed approaches using single compression techniques. Li et al. [26] presented a compression framework for conditional GANs via intermediate feature distillation and automated channel reduction in a “once-for-all” manner. Fu et al. [12] performed computational resources constrained differential neural architecture search via the guidance of knowledge distillation. Wang et al. [35] combined three compression techniques: model distillation, channel pruning, and quantization, together with the minimax objective, into one unified optimization to form an end-to-end optimization framework.

However, most of the above methods are not customized for GAN-based SR. Besides, they have not yet addressed the memory intensive problem. The memory budget should still be met for efficient network structure design. Moreover, the mutual balance of the compressed generator and discriminator needs to be considered, which is crucial for stabilizing the GANs training process. Our work is to solve the above problems.

3. Proposed Approaches

3.1. Network Structure

In our network, we first develop a novel generator architecture by designing the multi-scale feature aggregation modules (MFAMs). We then introduce the light-weight PatchGAN discriminator to overcome artifacts and facilitate MFAGAN training. Meriting from the above structures, we then distill both generator and discriminator to achieve non-trivial super-generator and to apply NAS channel pruning on it. The overall proposed compression framework architecture is depicted in Figure 2.

3.2. Multi-Scale Feature Aggregation Network

For the generator, inspired by ESRGAN [36], VoVNet [25] and IMDN [18], we design a memory-efficient Multi-Scale Feature Aggregation Network (MFANet) shown in Figure 4. The proposed MFANet mainly contains four parts: the coarse feature extraction part, the multi-scale feature extraction part, the feature fusion part, and the reconstruction part. In particular, we use a $3 \times 3$ convolution as the coarse feature extraction part to generate features from the input LR image. The following is the multi-scale features extraction part, in which three Multi-scale Feature Aggregation Modules (MFAMs) are stacked in a chain manner to refine the extracted features gradually. Later we will provide a detailed description of MFAM. After extracting multi-scale features with a set of MFAMs, we further conduct global
Figure 3: Illustration of three structures. (a) RFDB: the residual feature distillation block. (b) MFAM: the multi-scale feature aggregation module. (c) CFAB: the concatenative feature aggregation block in MFAM.

feature aggregation by a $1 \times 1$ convolution layer, which contributes to concatenate multi-scale features of different modules and reduces computation complexity. Meanwhile, a global skip connection is applied between different scale features so that the features information can be fully exploited. Then, a $3 \times 3$ convolution layer is used to smooth the aggregated features. Finally, the HR images are generated by the image reconstruction part, which only consists of a $3 \times 3$ convolution and a sub-pixel operation.

Modifying from the RFDN [27], as illustrated Figure 3 (a), we propose the Multi-scale Feature Aggregation Module (MFAM), as shown in Figure 3 (b), which is more expressive and memory-efficient than the RFDB. In RFDN, we can see that the channel reduction is conducted by a $1 \times 1$ convolution on the left, which compresses feature channels at a fixed ratio. Although the $1 \times 1$ convolution decreases the number of parameters, it increases the overall memory complexity. Therefore, the three $1 \times 1$ convolutions are removed, since we find that the NAS method is more efficient for channel reduction without introducing extra computation. Moreover, we also introduce the Concatenative Feature Aggregation Block (CFAB), as displayed in Figure 3 (c), to replace SRB in RFDN [27]. CFAB consists of five $3 \times 3$ convolutions, a local feature aggregation layer, a CCA layer [18], a $1 \times 1$ convolution layer, and a local skip connection. CFAB can conserve feature representations of multiple receptive fields as well as preserve original information. This schema provides diverse information for recovering high-resolution details. In summary, the main goal of MFAM is to reduce memory overhead and enhance expressive ability.

3.3. PatchGAN Discriminator

Besides the memory-efficient generator, the ESRGAN discriminator is replaced with the PatchGAN discriminator [29]. We utilize a 7-layer fully convolutional discriminator. Each convolutional layer is followed by a leaky ReLU. To avoid unpleasant artifacts, all BN layers are removed. Compared with the original discriminator, the PatchGAN discriminator has fewer parameters. Another advantage is that it only models local patches instead of the whole image, making the MFAGAN training faster and more stable.

3.4. Knowledge Distillation

Layer-wise knowledge from the teacher generator. Several attempts have been made to compress GANs generator with knowledge distillation in image translation. In this work, we match the teacher generator’s intermediate representations, as the layer-wise knowledge distillation work in [29]. In particular, we first train a teacher generator $G$ until convergence and then conduct layer-wise knowledge transfer from $G$ to the student generator $G'$. The goal of distillation is that the feature maps of each layer in $G'$ should be as close as possible to those of $G$. Feature maps of the 1st, 2nd and 3rd MFAMs of $G$ are denoted as $t_1$, $t_2$ and $t_3$ respectively. The corresponding levels of feature maps in $G'$ are the outputs of the 1st, 2nd, and 3rd MFAMs, denoting as $s_1$, $s_2$, and $s_3$, respectively. To address the above issue, we use the information in $t_1$, $t_2$ and $t_3$ to guide the information $s_1$, $s_2$ and $s_3$ during the training of $G'$. Subsequently, a student generator of fewer channels is trained by inheriting the low-level and high-level information from the original heavy teacher generator.

Layer-wise knowledge from the teacher discriminator. Although we aim to compress the generator, a discriminator stores useful knowledge of a learned GAN to guide the training generator [7]. Using heavy discriminator, $D$ sometimes leads to severe training instability and image quality degradation after the generator $G$ is compressed. It is necessary to distill the teacher discriminator $D$ to assist the training of the compressed generator $G'$. In this work, we adopt the PatchGAN discriminator architecture to distill layer-wise knowledge from teacher discriminator $D$ to student discriminator $D'$. Concretely, we extract corresponding levels of feature maps using the outputs of every two convolution layers, denoting as $t_2$, $t_4$, $t_6$, respectively. After that, we use the information in $t_2$, $t_4$, $t_6$ to guide the information $s_2$, $s_4$, $s_6$ during training of student discriminator. We jointly optimize $G'$ and $D'$ to minimize the distillation loss $L_{\text{Distill}_G}$ and $L_{\text{Distill}_D}$.

Mode collapse frequently occurs when the generator and discriminator are imbalanced. We adopt the layer-wise knowledge distillation on both the teacher generator and teacher discriminator. Hence, the student generator and discriminator are better matched. Such guidance from the teacher networks provides stable supervision in the early training phase of compressed networks. It is easy to implement, and mode collapse has never been experienced with our knowledge distillation schema.
3.5. Objective Function

**Overall objective.** There are five loss functions applied to training, depicted in Figure 4. The overall objective for our MFAGAN is the weighted sum of loss terms, written as follows:

$$L = \lambda_1 L_{\text{recon}} + \lambda_2 L_{\text{Distill}G} + \lambda_3 L_{\text{Distill}D} + \lambda_4 L_{\text{percep}} + \lambda_5 L_G,$$

(1)

where $\lambda_1$, $\lambda_2$, $\lambda_3$, $\lambda_4$, and $\lambda_5$ are the trade-off hyper-parameters to balance different objectives.

**Reconstruction loss.** Here, we apply the reconstruction loss, specifically the Mean Absolute Error (MAE) loss, to enhance the fidelity of the recovered images. Reconstruction loss is trained to optimize the $L1$ distance between the recovered images and ground-truths:

$$L_{\text{recon}} = \frac{1}{N} \sum_{i=1}^{N} \| R(x_i^{\text{lr}}) - x_i^{\text{gt}} \|_1,$$

(2)

where $x_i^{\text{lr}}$, $x_i^{\text{gt}}$ denote the $i$-th LR image patch and the corresponding HR. $N$ is the total number of training samples. $R(\cdot)$ represents the super-resolved output by MFAGAN.

**Layer-wise knowledge distillation loss.** We introduce a layer-wise knowledge distillation to extract the intermediate feature maps of the teacher generator. The intermediate feature maps contain richer information and allow the student generator to acquire low-level and high-level information from the teacher generator. The generator distillation objective can be formalized as:

$$L_{\text{Distill}G} = \frac{1}{n} \sum_{i=1}^{n} \| G_i(x) - G'_i(x) \|_2,$$

(3)

where $G_i(x)$ and $G'_i(x)$ are the intermediate feature maps of the $i$-th chosen layer in the teacher and student generator. Besides the generator, the discriminator stores useful knowledge of a learned GAN-based SR. It is useful to distill the teacher discriminator to stabilize the compressed generator training. The discriminator distillation loss function can be defined as:

$$L_{\text{Distill}D} = \frac{1}{m} \sum_{i=1}^{m} \| D_i(y) - D'_i(y) \|_2,$$

(4)

where $D_i(y)$ and $D'_i(y)$ are the feature maps of the $i$-th chosen layer in the teacher and student discriminator.

**Perceptual loss.** In [20] Johnson et al. proposed the perceptual loss to improve the visual effect of low-frequency features such as edges. Instead of computing distances in image pixel space, the images are first mapped into feature space by a pre-trained VGG19 network, denoted as $\phi$, and then compute the Mean Square Error (MSE) on their feature maps as follows:

$$L_{\text{percep}} = \sum_{i=1}^{n} \| \phi(\hat{y}_i) - \phi(y_i) \|_2,$$

(5)

where $\phi(\hat{y}_i)$ and $\phi(y_i)$ represent the feature maps of HR ground truth and the SR, respectively. All the feature maps are obtained by the fourth convolutional layer before the fifth max-pooling layer within the VGG19 network.

**Adversarial loss.** Following common practice, we apply adversarial loss [36] to enhance the texture details of the generator generated image to make it more realistic. Adversarial training a standard minimax optimization, and the discriminator $D$ is trained to distinguish between real images and the output of $G$. The adversarial loss $L_G$ is described as:

$$L_G = -\log(D(G(I^{LR}))),$$

(6)

where $I^{LR}$ is the LR image, $D(G(I^{LR}))$ means the probability of the discriminator over all training samples.

3.6. Hardware-aware NAS based Channel Pruning

**SuperGenerator training with fine-grained channels.** To address the fine-grained channel pruning problem, we first build a SuperGenerator that comprises all candidate SubGenerators. Concretely, we use MFANet as the backbone to build a “once-for-all” [6] network that comprises many SubGenerators of different channel numbers (i.e., 48, 32, 24), in which the full-width model is the SuperGenerator. The combined search space contains about $3^8 = 6581$...
different SubGenerators, in which every SubGenerator in the search space is a part of the SuperGenerator. In practice, the SuperGenerator only needs to be trained for the same steps as a baseline SR model, which is fast and low-cost. We thus use the most important channels of the SuperGenerator to initialize the SubGenerators. All SubGenerators share the front portion of corresponding layer weights in the SuperGenerator.

**Hardware-aware evolutionary search for specialized SubGenerator.** After SuperGenerator training, we adopt the evolutionary search to find the satisfactory SubGenerator, which satisfies the target hardware’s latency constraints while optimizing the PSNR. We can first build a lookup table containing the latency for all possible operators on the target hardware, then the overall latency of a SubGenerator is predicted by summing up each operator’s latency \[\sum_{i=1}^{N} \lambda_i \times \text{Latency}_i\]. Therefore, we can approximate the latency of candidate SubGenerator by querying the lookup table. The PSNR of SubGenerators is evaluated on the validation set. Afterward, we conduct the evolutionary search to get a specialized SubGenerator. Since SubGenerators training has been decoupled from the architecture search, we do not need any training cost in the search stage. This hardware-aware NAS channel pruning enables us to design a specialized SubGenerator on the target hardware.

Finally, we can fine-tune the pruned SubGenerator with distilled discriminator to obtain the final model.

4. Experimental Results

4.1. Experimental Setup

**Datasets.** Following [35], we adopt 800 HR images from the DIV2K dataset [2] as the training set. To generate LR training patches, we down-sample the HR images using bicubic interpolation in MATLAB. We also augment the training data with the random crop, horizontal/vertical flips, and $90^\circ$ rotations.

**Evaluation metrics.** For evaluation, we test the compressed/searched model on four SR benchmark datasets, namely Set5 [5], Set14 [38], B100 [33], and Urban100 [17]. Inspired by the PIRM2018-SR Challenge [34], we introduce PSNR and LPIPS on the Y channel of the transformed YCbCr space as the quality evaluation metrics. To measure the computation efficiency, we compare the widely used metrics - memory access cost, parameters, FLOPs (floating-point operations), and inference latency. Regarding the inference latency, we use the published codes of competitors to evaluate on a server with 4.2GHz Intel i7 CPU, 32GB RAM, and an Nvidia V100 GPU card.

**Implement details.** The training process is divided into four main stages. (1) Constructing the MFAGAN$_L$ model. We first train an MFANet with the $L_1$ loss, while the learning rate is $2 \times 10^{-4}$ and 500K iterations. The MFANet initialized generator is then trained using the loss function in Eq. (1) with $\lambda_1 = 1$, $\lambda_4 = 1$, $\lambda_5 = 10$, the learning rate is initialized to $1 \times 10^{-4}$ and halved at $[5k, 10k]$ iterations. (2) Layer-wise distillation on both the generator and discriminator in the MFAGAN$_L$. The student generator and student discriminator is trained with $\lambda_1 = 1$, $\lambda_2 = 0.05$, $\lambda_3 = 0.05$, $\lambda_4 = 1$, $\lambda_5 = 10$. (3) Training the weight-shared SuperGenerator. SuperGenerator is trained with $\lambda_3 = 1$, $\lambda_4 = 1$, the learning rate is set to $1 \times 10^{-4}$ and halved at $[200k, 400k, 600k]$ iterations, (4) Fine-tuning the searched generator with distilled discriminator about 10K iterations. For all experiments, we use the Adam [22] optimization method with $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ to train all of the models. The mini-batch size is set to 16. Our networks are implemented using the PyTorch framework on 8 NVIDIA V100 GPUs. The entire training process takes about 120 GPU hours.

4.2. Model Complexity Analysis

To construct a memory-efficient SR model, the memory access cost of the network is vital. As discussed in previous sections, the proposed MFAGAN could significantly reduce memory consumption. Figure 5 depicts the comparisons about LPIPS vs. memory access cost and FLOPs on Set5 4× dataset. From Figure 5, we can observe that the MFAGAN model with 3 MFAMs exhibits comparative or better performance and fewer memory usage than other state-of-the-art methods SRGAN [23], EnhanceNet [30], ESRGAN [36], and AGD [12]. MFAGAN is also superior over efficient models including SRCNN [10], IMDN [18]. Compared with IMDN, our MFAGAN achieves better LPIPS with a slightly larger model. These results demonstrate that the proposed MFAGAN can correctly balance memory complexity and reconstruction performance.

4.3. Ablation Study

In this section, we conduct ablation experiments to investigate the contributions of each component in the proposed method. The overall comparison is illustrated in Table 1 in

Figure 5: The memory access cost vs. LPIPS on Set5 (4×) dataset. The orange circles represent our proposed models. The circles’ size represents the number of FLOPs, which are calculated on 512×512 HR image.
which each column represents a model. A detailed discussion is provided as follows.

Table 1: Ablation study: Memory-efficient architecture combined with knowledge distillation on both $G$ and $D$, and NAS channel pruning achieves the best performance on the Set5 dataset.

| Options               | Baseline (ESRGAN) | 1st | 2nd | 3rd | 4th | 5th |
|-----------------------|-------------------|-----|-----|-----|-----|-----|
| generator channels    | 64                | 64  | 48  | 48  | 32  | 32  |
| discriminator channels| 64                | 48  | 32  | 48  | 48  | 32  |
| MFAGAN$_L$            | ✓                 | ✓   | ✓   | ✓   | ✓   | ✓   |
| Distill $G$ and $D$    |                   | ✓   | ✓   | ✓   | ✓   | ✓   |
| Distill $G$           |                   |     |     |     |     |     |
| NAS pruning           |                   | ✓   | ✓   | ✓   | ✓   | ✓   |
| PSNR$^+$              | 30.45             | 30.32 | 30.65 | 28.56 | 29.19 | 30.16 |
| LPIPS$^-$             | 0.0572            | 0.0527 | 0.0558 | 0.0870 | 0.0672 | 0.0571 |
| Memory (G)            | 4.29              | 0.877 | 0.657 | 0.657 | 0.515 | 0.515 |
| #Param. (MB)          | 16.67             | 1.56  | 0.884 | 0.884 | 0.551 | 0.551 |
| #FLOPs (G)            | 291               | 23.72 | 13.45 | 13.45 | 8.41  | 8.41  |

Effectiveness of memory-efficient architecture. We first analyze the advantage of MFANet based MFAGAN$_L$. As shown in the 1st column, MFAGAN$_L$ has comparative PSNR and LPIPS results with the baseline model. While achieves $4.89 \times$ memory saving and $12.26 \times$ computation reduction. With our memory consumption and computation complexity largely reduced, the SR performance remains relatively stable. The perceptual SR task requires the model to be expressive enough to recover more realistic texture details. Our proposed multi-scale feature aggregation modules (MFAMs) is capable of aggregating multi-scale features to produce powerful feature representation.

Effectiveness of layer-wise knowledge distillation on both $G$ and $D$. We also investigate the effects of different distillation objectives on the MFAGAN$_L$. Two distillation methods are proposed, including solely layer-wise distillation on teacher generator $G$, and knowledge distillation on both teacher generator $G$ and teacher discriminator $D$. Results are illustrated in the 2nd and 3rd columns of Table 1, which show that the proposed distillation objective is useful for the MFAGAN$_L$ compression. As shown in the 2nd column, distilled MFAGAN$_L$ even has better PSNR result than MFAGAN$_L$, with $1.33 \times$ memory saving and $1.76 \times$ computation reduction. While solely distillation on $G$ yields worse performance compared with MFAGAN$_L$. As a teacher discriminator, $D$ stores useful information about teacher generator $G$. It can offer strong supervision to guide the student generator $G'$ to learn faster and better.

Effectiveness of NAS channel pruning. We further explore the role of NAS channel pruning. The results are shown in the 4th and 5th columns of Table 1. Directly using NAS channel pruning to compress the MFAGAN$_L$ generator largely degrades the image SR performance. In contrast, knowledge distillation + NAS channel pruning achieves much better PSNR and LPIPS results, showing the necessity of jointly using channel pruning and knowledge distillation. Actually, the capacity gap between the SuperGenerator, i.e., MFAGAN-64, and the directly searched SubGenerator, i.e., MFAGAN-32, are too huge. As a result, the inherited weights from the SuperGenerator may be too recondite for the SubGenerator, in which case large ratio NAS channel pruning would have negative effects on the searched model. Applying the distilled MFAGAN-48 as SuperGenerator allows us to find a SubGenerator, which has a smaller gap between the SuperGenerator and hence makes learning easier. NAS channel pruning and knowledge distillation are complementary to each other, which guarantees MFAGAN achieve competitive results.

Effectiveness of discriminator. Finally, this section aims to evaluate the importance of the PatchGAN discriminator. To this end, we train the MFAGAN$_L$ with PatchGAN discriminator and ESRGAN discriminator on DIV2K, respectively. The convergence curves are visualized in Figure 6. We can observe that PatchGAN discriminator achieves better results, verifying that: (1) vanilla ESRGAN discriminator is not matched with light-weight MFANet and has worse performance; (2) PatchGAN discriminator leads to significantly better PSNR result and stability.

4.4. Comparison with State-of-the-Art Methods

Quantitative comparisons. We report the quantitative comparisons of state-of-the-art perceptual-driven methods over the benchmark datasets in Table 2. Compared with given methods, we can see MFAGAN achieves the best PSNR and LPIPS performance in most datasets. This reveals the effectiveness of our MFAM. Additionally, we also give the FLOPs, memory access cost, parameters, and latency for all the comparison methods. Our model achieves large compression ratios. In particular, our proposed method shows a clear advantage of ESRGAN compression compared to the previous GAN compression method AGD [12]. We can reduce the memory access cost of the ESR-
Table 2: Quantitative comparison of our model with state-of-the-art perceptual-driven works on $4 \times 4$ SR task. In each row, red/blue represents best/second, respectively.

| Model          | Memory (G) | #FLOPs (G) | #Param. (MB) | Latency (ms) | Set5 PSNR↑ | Set5 LPIPS↓ | Set14 PSNR↑ | Set14 LPIPS↓ | B100 PSNR↑ | B100 LPIPS↓ | Urban100 PSNR↑ | Urban100 LPIPS↓ |
|----------------|------------|------------|-------------|--------------|------------|-------------|------------|-------------|------------|-------------|----------------|----------------|
| SRGAN          | 0.7        | 36.5       | 1.55        | 80.8         | 29.40      | 0.0621      | 26.11      | 0.1167      | 25.17      | 0.1333      | –               | –               |
| EnhanceNet     | 0.91       | 30.2       | 0.85        | 29           | 28.56      | 0.0764      | 25.77      | 0.1295      | 24.93      | 0.1481      | 23.54          | 0.1307         |
| ESRGAN         | 4.29       | 291        | 16.67       | 169.4        | 30.45      | 0.0572      | 26.28      | 0.1055      | 25.32      | 0.1216      | 24.36          | 0.1000         |
| AGD            | 0.94       | 27.1       | 0.41        | 27.7         | 30.41      | 0.0700      | 27.27      | 0.1247      | 26.22      | 0.1556      | 24.73          | 0.1329         |
| MFAGAN (ours)  | 0.52       | 8.41       | 0.55        | 21.9         | 30.16      | 0.0571      | 26.69      | 0.1133      | 25.33      | 0.1332      | 24.23          | 0.1132         |

GAN generator by 8.2×, which is 2× better compared to the previous AGD method while achieving a much better PSNR and LPIPS. It demonstrates that MFAGAN is superior to other perceptual-driven methods in a comprehensive performance.

Table 3: Inference latency comparison of our MFAGAN with ESRGAN on the Qualcomm Snapdragon 865 GPU on $4 \times 4$ SR task.

| Model          | Memory (G) | #FLOPs (G) | #Param. (MB) | Set5 PSNR↑ | Set5 LPIPS↓ | Mobile Latency |
|----------------|------------|------------|-------------|------------|-------------|----------------|
| ESRGAN         | 4.29       | 291        | 16.67       | 30.45      | 0.0572      | 1150           |
| MFAGAN (ours)  | 0.52       | 8.41       | 0.55        | 30.16      | 0.0571      | 70             |

4× Qualitative comparisons. To further illustrate the analysis above, we show visual comparisons on scales 4× on B100 and Set14. From Figure 7, it can be observed that MFAGAN can generate sharp edges and realistic textures without introducing unpleasant artifacts. For image “8023” and “223061” in the B100 dataset, we can see that MFAGAN can recover the sharpness of the edges on the objects. Besides, for challenging details in the image “Lena” in Set14, MFAGAN can generate the correct textures of hair portion and hat edges. In general, MFAGAN achieves comparably visual quality with ESRGAN and shows more realistic textures and sharper edges over IMDN and AGD.

4.5. Comparison of Latency on Mobile Device

Mobile inference acceleration has drawn people’s attention in recent years. At last, we compare the inference latency of our MFAGAN with ESRGAN on Qualcomm Snapdragon 865 GPU (on OPPO Find X2) for 4× SR images. We use the TensorFlow Lite framework to deploy the models on the mobile phone. The results are reported in Table 3. We can see that our mobile-friendly model can process a 128×128 input with 70 ms, while ESRGAN with comparable performance requires a significantly longer 1150ms. Our proposed MFAGAN achieves 16.4× measured speedup while saving memory usage by 88%. The results show that MFAGAN is highly efficient in real-world applications. The algorithm and hardware co-design enables us to design specialized models on the target hardware.

5. Conclusion

In this work, we propose Multi-scale Feature Aggregation Net based GAN (MFAGAN) compression framework to reduce the memory consumption of the generator in GAN-based SR. MFAGAN leverages memory-efficient architecture, layer-wise knowledge distillation, and hardware-aware evolutionary search to stabilize the training and improve the memory-efficiency. Extensive experiments show MFAGAN outperforms previous state-of-the-art methods with aggressively reduced memory access cost and a faster inference speed without visual quality degradation. For future works, we have plans...
to apply our findings to video SR.

References

[1] Angeline Aguinaldo, Png-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. arXiv preprint arXiv:1902.00159, 2019.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 126–135, 2017.

[3] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In Proceedings of the European Conference on Computer Vision (ECCV), pages 252–268, 2018.

[4] Saeed Anwar and Nick Barnes. Densely residual laplacian super-resolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

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

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791, 2019.

[7] Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunqing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. arXiv preprint arXiv:2003.03519, 2020.

[8] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. arXiv preprint arXiv:1901.07261, 2019.

[9] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks. arXiv preprint arXiv:1803.07422, 2018.

[10] 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.

[11] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In European conference on computer vision, pages 391–407. Springer, 2016.

[12] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. arXiv preprint arXiv:2006.08198, 2020.

[13] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 3224–3234, 2019.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[15] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3):243–254, 2016.

[16] 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.

[17] 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.

[18] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In Proceedings of the 27th ACM International Conference on Multimedia, pages 2024–2032, 2019.

[19] 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.

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European conference on computer vision, pages 694–711. Springer, 2016.

[21] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recurrent convolutional network for image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1637–1645, 2016.

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

[23] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4681–4690, 2017.

[24] Juhyun Lee, Nikolay Chirkov, Ekaterina Ignashova, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. arXiv preprint arXiv:1907.01989, 2019.

[25] Youngwan Lee, Joon-ho Hwang, Sangrok Lee, Yueseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019.

[26] Muyang Li, Ji Lin, Yaoqiao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5284–5294, 2020.
[27] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. *arXiv preprint arXiv:2009.11551*, 2020.

[28] André Lopes, Frederico Pratas, Leonel Sousa, and Aleksandar Ilic. Exploring gpu performance, power and energy-efficiency bounds with cache-aware roofline modeling. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 259–268. IEEE, 2017.

[29] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[30] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4491–4500, 2017.

[31] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[32] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunqing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3235–3244, 2019.

[33] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian conference on computer vision*, pages 111–126. Springer, 2014.

[34] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. Ntire 2018 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 852–863, 2018.

[35] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *European Conference on Computer Vision*, pages 54–73. Springer, 2020.

[36] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[37] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

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

[39] Kai Zhang, Martin Danelljan, Yawei Li, Radu Timofte, Jie Liu, Jie Tang, Gangshan Wu, Yu Zhu, Xiangyu He, Wenjie Xu, et al. Aim 2020 challenge on efficient super-resolution: Methods and results. *arXiv preprint arXiv:2009.06943*, 2020.

[40] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3096–3105, 2019.

[41] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

[42] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018.

[43] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.