2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Fast, Accurate and Lightweight Super-Resolution with Neural Architecture Search

Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu
Xiaomi AI Lab
Beijing, China
Email: {chuxiangxiang,zhangbo11,mahailong,xuruijun}@xiaomi.com

Qingyuan Li
Xiaomi IoT
Beijing, China
Email: liqingyuan@xiaomi.com

***Abstract*—Deep convolutional neural networks demonstrate impressive results in the super-resolution domain. A series of studies concentrate on improving peak signal noise ratio (PSNR) by using much deeper layers, which are not friendly to constrained resources. Pursuing a trade-off between the restoration capacity and the simplicity of models is still non-trivial. Recent contributions are struggling to manually maximize this balance, while our work achieves the same goal automatically with neural architecture search. Specifically, we handle super-resolution with a multi-objective approach. We also propose an elastic search tactic at both micro and macro level, based on a hybrid controller that profits from evolutionary computation and reinforcement learning. Quantitative experiments help us to draw a conclusion that our generated models dominate most of the state-of-the-art methods with respect to the individual FLOPS.**

I. Introduction and Related Work

As a classical task in computer vision, single image super-resolution (SISR) is aimed to restore a high-resolution image from a degraded low-resolution one, which is known as an ill-posed inverse procedure. Most of the recent works on SISR have shifted their approaches to deep learning, and they have surpassed other SISR algorithms with big margins [1], [2], [3], [4].

Nonetheless, these human-designed models are tenuous to fine-tune or to compress. Meantime, neural architecture search has produced dominating models in classification tasks [5], [6]. Following this trend, a novel work by [7] has shed light on the SISR task with a reinforced evolutionary search method, which has achieved results outperforming some notable networks including VDSR [2]. We are distinct to [7] by stepping forward to design a dense search space which allows searching in both macro and micro level, which has led to significantly better visual results.

In this paper, we dive deeper into the SISR task with elastic neural architecture search, hitting a record comparable to CARN and CARN-M [4] [1]. Our main contributions can be summarized in the following four aspects,

- releasing several fast, accurate and lightweight super-resolution architectures and models (FALSR-A being the best regarding visual effects), which are highly competitive with recent state-of-the-art methods,
- performing elastic search by combining micro and macro space on the cell-level to boost capacity,
- building super-resolution as a constrained multi-objective optimization problem and applying a hybrid model generation method to balance exploration and exploitation,
- producing high-quality models that can meet various requirements under given constraints within a single run.

[1]Our models are released at https://github.com/falsr/FALSR.

II. Pipeline Architecture

Like most of Neural Architecture Search (NAS) approaches, our pipeline contains three principle ingredients: an elastic search space, a hybrid model generator and a model evaluator based on incomplete training. It is explained in detail in the following sections.

Similar to [8], [7], we also apply NSGA-II [9] to solve the multi-objective problem. Our work differs from them by using a hybrid controller and a cell-based elastic search space that enables both macro and micro search.

We take three objectives into account for the super-resolution task,

- quantitative metric to reflect the performance of models (PSNR),
- quantitative metric to evaluate the computational cost of each model (mult-adds),
- number of parameters.

In addition, we consider the following constraints,

- minimal PSNR for practical visual perception,
- maximal mult-adds regarding resource limits.

III. Elastic Search Space

Our search space is designed to perform both micro and macro search. The former is used to choose promising cells within each cell block, which can be viewed as a feature extraction selector. In contrast, the latter is aimed to search backbone connections for different cell blocks, which plays a role of combining features at selected levels. In addition, we use one cell block as our minimum search element for two reasons: design flexibility, and broad representational capacity.

Typically, the super-resolution task can be divided into three sub-procedures: feature extraction, nonlinear mapping, and restoration. Since most of the deep learning approaches concentrate on the second part, we design our search space to describe the mapping while fixing others. Figure 1 depicts our main flow for super-resolution. Thus, a complete model

978-1-7281-8808-9/20/$31.00 ©2020 IEEE

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9413080

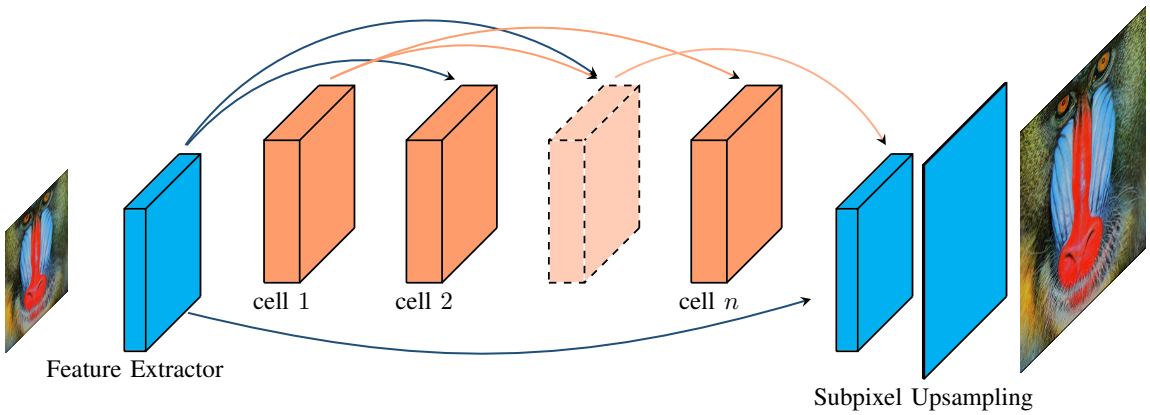

Fig. 1: Neural Architecture of Super-Resolution (the arrows denote skip connections).

contains a predefined feature extractor (a 2D convolution with 32 3 $\times$ 3 filters), n cell blocks drawn from the micro search space which are joined by the connections from macro search space, and subpixel-based upsampling and restoration[2].

A. Cell-Level Micro Search Space

For simplicity, all cell blocks share the same cell search space S. In specific, the micro search space comprises the following elements:

- convolutions: 2D convolution, grouped convolution with groups in $\{2, 4\}$, inverted bottleneck block with an expansion rate of 2,
- channels: $\{16, 32, 48, 64\}$,
- kernels: $\{1, 3\}$,
- in-cell residual connections:$\{\text{True}, \text{False}\}$,
- repeated blocks:$\{1, 2, 4\}$.

Therefore, the size of micro space for n cell blocks is 192^n.

B. Intercell Macro Search Space

The macro search space defines the connections among different cell blocks. Specifically, for the i-th cell block CB_i, there are $n + 1 - i$ choices of connections to build the information flow from the input of CB_i to its following cell blocks[3]. Furthermore, we use c_i^j to represent the path from input of CB_i to CB_j. We set $c_i^j = 1$ if there is a connection path between them, otherwise 0. Therefore, the size of macro space for n cell blocks is $2^{n(n+1)/2}$. In summary, the size of the total space is $192^n \times 2^{n(n+1)/2}$.

IV. Model Generator

Our model generator is a hybrid controller involving both reinforcement learning (RL) and an evolutionary algorithm (EA). The EA part handles the iteration process and RL is used to bring exploitation. To be specific, the iteration is controlled by NSGA-II [9], which contains four sub-procedures: population initialization, selection, crossover, and mutation. To avoid verbosity, we only cover our variations to NSGA-II.

[2]Our upsampling contains a 2D convolution with 32 3×3 filters, followed by a 3 $\times$ 3 convolution with one filter of unit stride.

[3]Here, i starts with 1.

A. Model Meta Encoding

One model is denoted by two parts: forward-connected cells and their information connections. We use the indices of operators from the operator set to encode the cells, and a nested list to depict the connections. Namely, given a model M with n cells, its corresponding chromosome can be depicted by (M_{mic}, M_{mac}), where M_{mic} and M_{mac} are defined as follows,

$$M_{mic} = (x_1, x_2, ..., x_n) \tag{1}$$

$$\begin{aligned} M_{mac} &= (c_1^{1:n}, c_2^{2:n}, ..., c_n^n) \\ c_i^{i:n} &= (c_i^i, c_i^{i+1}, ..c_i^n) \end{aligned} \tag{2}$$

B. Initialization

We begin with N populations and we emphasize the diversities of cells. In effect, to generate a model, we randomly sample a cell from S and repeat it for n times. In case N is larger than the size of S, models are arbitrarily sampled without repeating cells.

As for connections, we sample from a categorical distribution. While in each category, we pick uniformly, i.e. $p \sim \mathcal{U}(0, 1)$. To formalize, the connections are built based on the following rules,

$$\begin{cases} \text{random connections} & 0 \le p < p_r \\ \text{dense connections} & p_r \le p < p_r + p_{den} \\ \text{no connnections} & p_r + p_{den} \le p < 1 \end{cases} \tag{3}$$

C. Tournament Selection

We calculate the crowding distance as noted in [10] to render a uniform distribution of our models, and we apply tournament selection ($k = 2$) to control the evolution pressure.

D. Crossover

To encourage exploration, single-point crossovers are performed simultaneously in both micro and macro space. Given two models A $(M_{mic(A)}, M_{mac(A)})$ and B $(M_{mic(B)}, M_{mac(B)})$, a new chromosome C can be generated as,

$$\begin{aligned} M_{mic(C)} &= (x_{1A}, x_{2A}, ..., x_{iB}, ..., x_{nA}) \\ M_{mac(C)} &= (c_{1A}^{1:n}, c_{2A}^{2:n}, ..., c_{jB}^{j:n}, ..., c_{nA}^n) \end{aligned} \tag{4}$$

where i and j are chosen positions respectively for micro and macro genes. Informally, the crossover procedure contributes more to exploitation than to exploration.

E. Mutation

We again apply a categorical distribution to balance exploration and exploitation.

1) Exploration: To encourage exploration, we combine random mutation with roulette wheel selection (RWS). Since we treat super-resolution as a multi-objective problem, FLOPS and the number of parameters are two objectives that can be evaluated soon after meta encodings are available. In particular, we also sample from a categorical distribution to determine mutation strategies, i.e. random mutation (with an upper-bound probability p_{mr}) or mutated by roulette wheel selection to handle FLOPS (lower than p_{mf}) or parameters. Formally,

$$\begin{cases} \text{random mutation} & 0 \le p < p_{mr} \\ \text{RWS for FLOPS} & p_{mr} \le p < p_{mf} \\ \text{RWS for params} & p_{mf} \le p < 1 \end{cases} \tag{5}$$

Whenever we need to mutate a model M by RWS, we keep M_{mac} unchanged. Since each cell shares the same operator set S, we perform RWS on S for n times to generate M_{mic}. Strictly speaking, given M_{mac}, it's intractable to execute a complete RWS (involving 192^n models). Instead, it can be approximated based on S (involving 192 basic operators). Besides, we scale FLOPS and the number of parameters logarithmically before RWS.

2) Exploitation: To enhance exploitation, we apply a reinforcement driven mutation.

We use a neural controller to mutate, which is shown in Figure 2. Specifically, the embedding features for M_{mic} are concatenated, and then are injected into 3 fully-connected layers to generate M_{mac}. The last layer has $n(n+1)/2$ neutrons to represent connections, with its output denoted as O^{mac}.

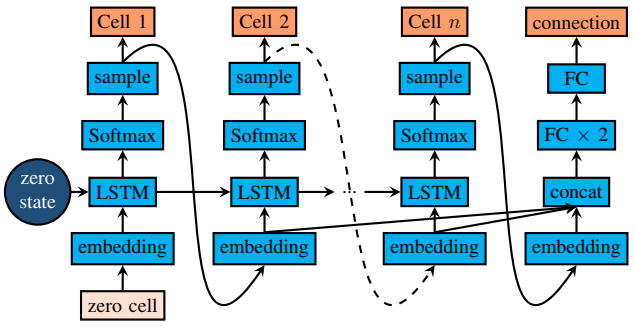

Fig. 2: The controller network to generate cells and connections.

The network parameters can be partitioned into two groups, θ^{mic} and θ^{mac}. The probability of selecting S_i for cell j is $p(cell_i = S_i|\theta^{mic})$ and for the connection $c_i^j = 1$, we have $p(c_i^j = 1|\theta^{mac}) = O^{mac}_{(i-1)*(n+1-0.5*i)+j}$. Thus, the gradient $g(\theta)$ can be calculated as follows:

$$\begin{aligned} g(\theta) = -\nabla_\theta [& \sum_{i=1}^{n} \log p(cell_i = S_i|\theta^{mic}) * R_i + \\ & \sum_{j=1}^{n(n+1)/2} c_j \log O_j^{mac} * R_j + \\ & (1 - c_j) \log(1 - O_j^{mac}) * R_j]. \end{aligned} \tag{6}$$

In Equation 6, R_i and R_j are the discounted accumulated rewards. Here, we set the discount parameter $\gamma = 1.0$.

V. Evaluator

The evaluator calculates the scores of the models generated by the controller. In the beginning, we attempted to train an RNN regressor to predict the performances of models, with data collected in previous pipeline execution. However, its validation error is too high to continue. Instead, each model is trained for a relatively short time (see the 'incomplete training' part in Section VI-A) to roughly differentiate various models. At the end of the incomplete training, we evaluate mean square errors on test datasets.

VI. Experiments

A. Setup

In our experiment, about 10k models are generated in total, where the population for each iteration is 64. The Pareto-front of all the models is shown in Fig. 6. It takes less than 3 days on a Tesla-V100 with 8 GPUs to execute the pipeline once. We use DIV2K as our training set.

During an incomplete training, each model is trained with a batch size of 16 for 200 epochs. In addition, we apply Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) to minimize the L_1 loss between the generated high-resolution images and its ground truth. The learning rate is initialized as 10^{-4} and kept unchanged at this stage.

As for the full train, we choose 4 models with a large crowding distance in the Pareto front between mean squared error and mult-adds, which was generated at the incomplete training stage. These models are trained based on DIV2K dataset for 24000 epochs with a batch-size of 16 and it takes less than 1.5 days. Moreover, the standard deviation of weights w is initialized as 0.02 and the bias 0.

B. Comparisons with State-of-the-Art Super-Resolution Methods

After being fully trained, our model are compared with the state-of-the-art methods on the commonly used test dataset for super-resolution (See Table I and Figure 5). To be fair, we only consider the models with comparable FLOPS. Therefore, too deep and large models such as RDN [17], RCAN [18] are excluded here. We choose PSNR and SSIM as metrics by convention [19]. The comparisons are made on the $\times 2$ task. Note that all mult-adds are measured based on a 480×480 input.

TABLE I: Comparisons with the state-of-the-art methods based on $\times 2$ super-resolution task.

Model	Mult-Adds	Params	SET5 PSNR/SSIM	SET14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
SRCNN [1]	52.7G	57K	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946
FSRCNN [11]	6.0G	12K	37.00/0.9558	32.63/0.9088	31.53/0.8920	29.88/0.9020
VDSR [2]	612.6G	665K	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140
DRCN [12]	17,974.3G	1,774K	37.63/0.9588	33.04/0.9118	31.85/0.8942	30.75/0.9133
LapSRN [13]	29.9G	813K	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100
DRRN [14]	6,796.9G	297K	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188
SelNet [15]	225.7G	974K	37.89/0.9598	33.61/0.9160	32.08/0.8984	-
CARN [4]	222.8G	1,592K	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256
CARN-M [4]	91.2G	412K	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9194
MoreMNAS-A [7]	238.6G	1,039K	37.63/0.9584	33.23/0.9138	31.95/0.8961	31.24/0.9187
AWSRN-M [16]	244.1G	1,063K	38.04/0.9605	33.66/0.9181	32.21/0.9000	32.23/0.9294
FALSR-A (ours)	234.7G	1,021K	37.82/0.9595	33.55/0.9168	32.12/0.8987	31.93/0.9256
FALSR-B (ours)	74.7G	326k	37.61/0.9585	33.29/0.9143	31.97/0.8967	31.28/0.9191
FALSR-C (ours)	93.7G	408k	37.66/0.9586	33.26/0.9140	31.96/0.8965	31.24/0.9187

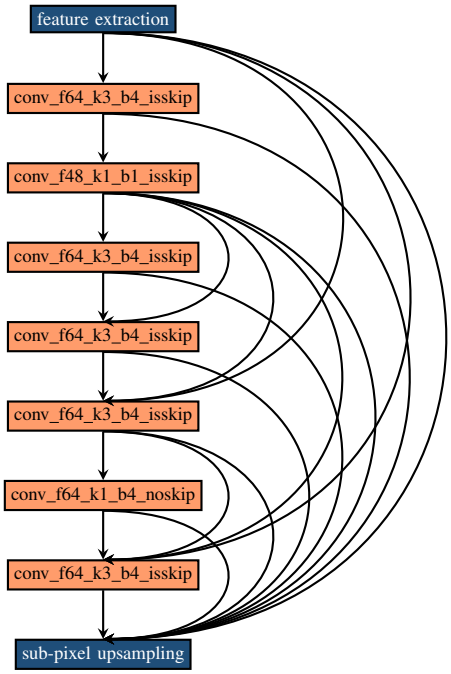

Fig. 3: The model FALSR-A (the one with best visual effects) comparable to CARN. Note for instance, 'conv_f64_k3_b4_isskip' represents a block of 4 convolution layers, each with a filter size of 64 and a kernel size of 3×3, including a skip connection to form residual structure.

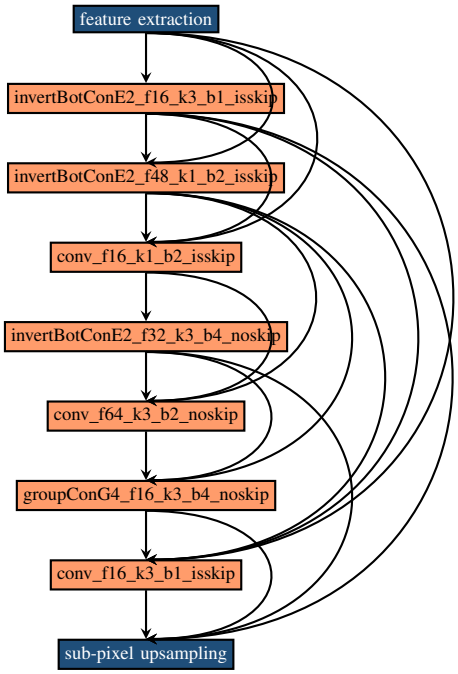

Fig. 4: The model FALSR-B comparable to CARN-M.

At a comparable level of FLOPS, our model called FALSR-A (Figure 3) outperforms CARN [4] with higher scores. In addition, it dominates DRCN [12] and MoreMNAS-A [7] over three objectives on four datasets. Moreover, it achieves higher PSNR and SSIM with fewer FLOPS than VDSR [2], DRRN [14] and many others.

For a more lightweight version, one model called FALSR-B (Figure 4) dominates CARN-M, which means with fewer FLOPS and a smaller number of parameters it scores equally to or higher than CARN-M. Besides, its architecture is attractive and the complexity of connections lies in between residual and dense connections. This means a dense connection is not always the optimal way to transmit information. Useless features from lower layers could make trouble for high layers to restore super-resolution results.

Another lightweight model called FALSR-C (not drawn because of space) also outperforms CARN-M. This model uses relatively sparse connections (8 in total). We conclude that this sparse flow works well with the selected cells.

Figure 7 shows the qualitative results against other methods.

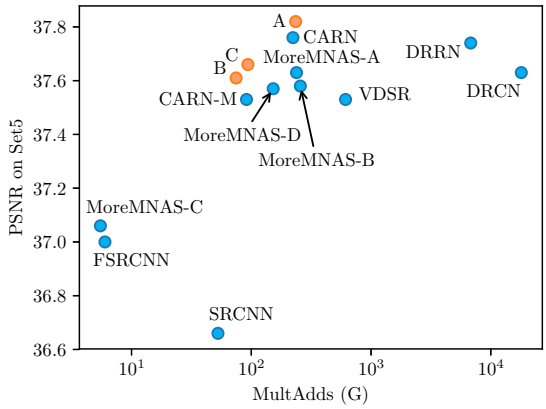

Fig. 5: FALSR A, B, C (shown in salmon) vs. others (light blue)

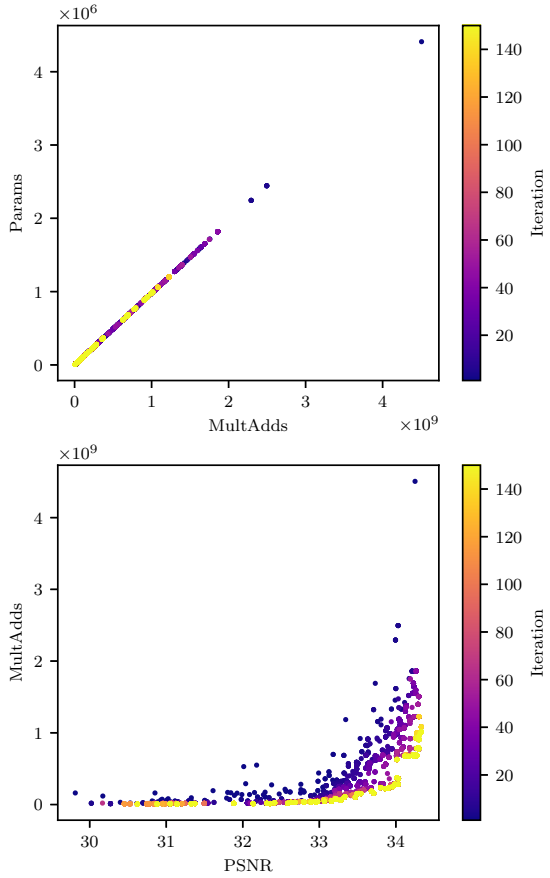

Fig. 6: The Pareto-front of of all the models during the evolution, paired every two objectives.

C. Discussions

1) Cell Diversity: Our experiments show that a good cell diversity also helps to achieve better results for super-resolution, same for classification tasks [20]. In fact, we have trained several models with repeated blocks, however, they underperform the models with diverse cells. We speculate that different types of cells can handle input features more effectively than monotonous ones.

2) Optimal Information Flow: Perhaps under given current technologies, dense connections are not optimal in most cases. In principle, a dense connection has the capacity to cover other non-dense configurations, however, it's usually difficult to train a model to ignore useless information.

3) Good Assumption?: Super-resolution is different from feature extraction domains such as classification, where more details need to be restored at pixel level. Therefore, it rarely applies downsampling operations to reduce the feature dimensions and it is more time-consuming than classification tasks like on CIFAR-10.

Regarding the time, we use incomplete training to differentiate various models. This strategy works well under an implicit assumption: models that perform better when fully trained also behave well with a large probability under an incomplete training. Luckily, most of deep learning tasks share this good feature. For the rest, we must train models as fully as possible.

VII. Conclusions

To sum up, we presented a novel elastic method for NAS that incorporates both micro and macro search, dealing with neural architectures in multi-granularity. The result is exciting as our generated models dominate the newest state-of-the-art SR methods. Different from human-designed and single-objective NAS models, our methods can generate different tastes of models by one run, ranging from fast and lightweight to relatively large and more accurate. Therefore, it offers a feasible way for engineers to compress existing popular human-designed models or to design various levels of architectures accordingly for constrained devices.

Our future work will focus on training a model regressor, which estimates the performance of models, to speed up the pipeline.

References

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

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

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and, lightweight super-resolution with cascading residual network," *arXiv preprint arXiv:1803.08664*, 2018.

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

[6] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *arXiv preprint arXiv:1707.07012*, vol. 2, no. 6, 2017.

[7] X. Chu, B. Zhang, R. Xu, and H. Ma, "Multi-objective reinforced evolution in mobile neural architecture search," *arXiv preprint arXiv:1901.01074*, 2019.

[8] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: A multi-objective genetic algorithm for neural architecture search," *arXiv preprint arXiv:1810.03522*, 2018.

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

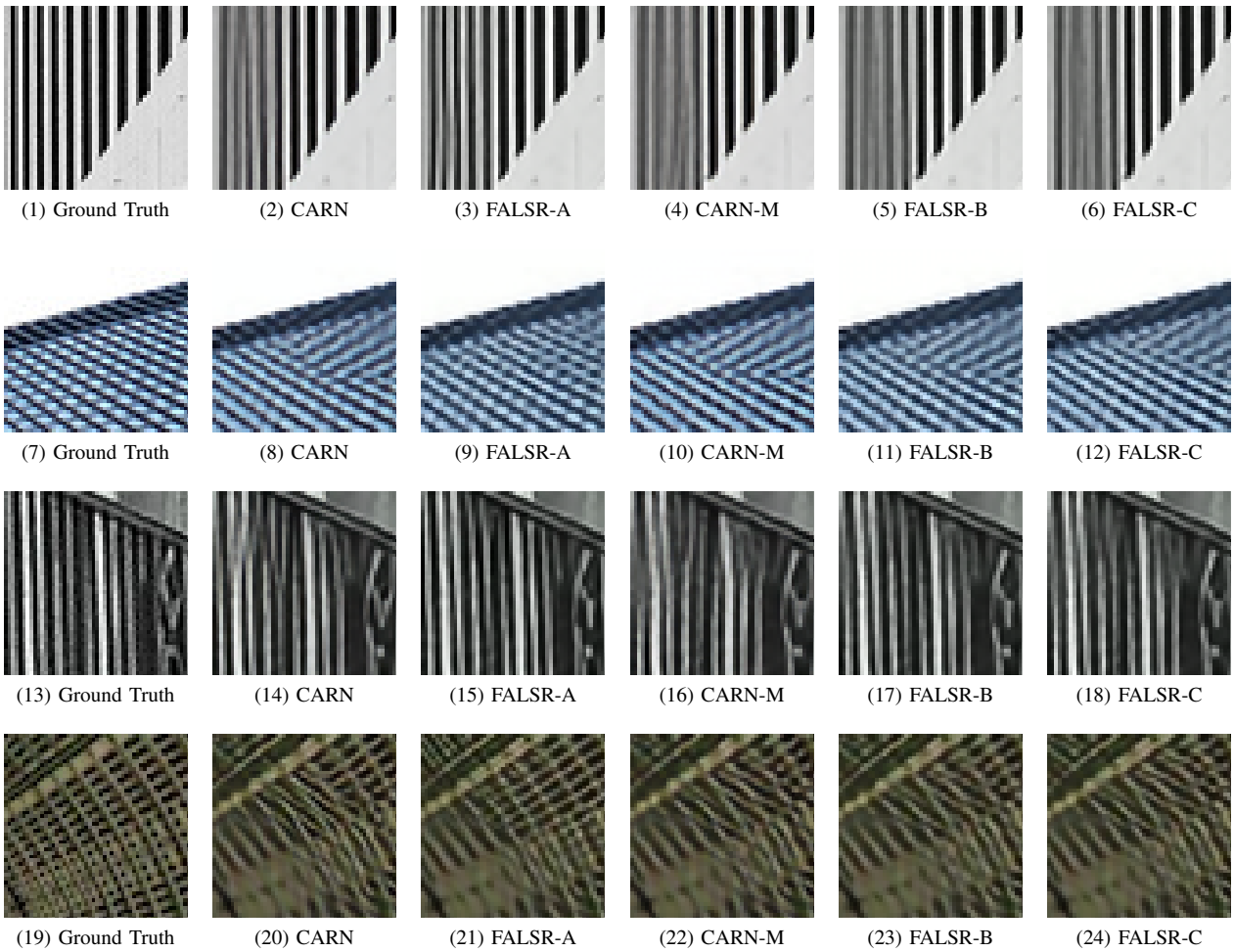

Fig. 7: Qualitative results on images from Urban100 (image ids in rows from top to bottom: 011, 062, 066, 078).

[10] X. Chu and X. Yu, "Improved crowding distance for nsga-ii," *arXiv preprint arXiv:1811.12667*, 2018.

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

[12] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1637–1645.

[13] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate superresolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 3, 2017, p. 5.

[14] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 5.

[15] J.-S. Choi and M. Kim, "A deep convolutional neural network with selection units for super-resolution," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 1150–1156.

[16] C. Wang, Z. Li, and J. Shi, "Lightweight image super-resolution with adaptive weighted learning network," *arXiv preprint arXiv:1904.02358*, 04 2019. [Online]. Available: https://arxiv.org/abs/1904.02358

[17] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision, Munich, Germany*, 2018, pp. 8–14.

[19] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Pattern recognition (icpr), 2010 20th international conference on*. IEEE, 2010, pp. 2366–2369.

[20] C.-H. Hsu, S.-H. Chang, D.-C. Juan, J.-Y. Pan, Y.-T. Chen, W. Wei, and S.-C. Chang, "Monas: Multi-objective neural architecture search using reinforcement learning," *arXiv preprint arXiv:1806.10332*, 2018.