Hitchhiker's Guide to Super-Resolution: Introduction and Recent Advances

Brian B. Moser, Federico Raue, Stanislav Frolov, Sebastian Palacio, Jörn Hees, and Andreas Dengel

(Survey Paper)

***Abstract*—With the advent of Deep Learning (DL), Super-Resolution (SR) has also become a thriving research area. However, despite promising results, the field still faces challenges that require further research, e.g., allowing flexible upsampling, more effective loss functions, and better evaluation metrics. We review the domain of SR in light of recent advances and examine state-of-the-art models such as diffusion (DDPM) and transformer-based SR models. We critically discuss contemporary strategies used in SR and identify promising yet unexplored research directions. We complement previous surveys by incorporating the latest developments in the field, such as uncertainty-driven losses, wavelet networks, neural architecture search, novel normalization methods, and the latest evaluation techniques. We also include several visualizations for the models and methods throughout each chapter to facilitate a global understanding of the trends in the field. This review ultimately aims at helping researchers to push the boundaries of DL applied to SR.**

***Index Terms*—Artificial intelligence, computer science, deep learning, IEEE, super-resolution, survey, TPAMI.**

I. Introduction

SUPER-RESOLUTION (SR) is the process of enhancing Low-Resolution (LR) images to High-Resolution (HR). The applications range from natural images [1], [2] to highly advanced satellite [3], and medical imaging [4]. Despite its long history [5], SR remains a challenging task in computer vision because it is notoriously ill-posed: several HR images can be valid for any given LR image due to many aspects like brightness and coloring [6], [7]. The fundamental uncertainties in the relation between LR and HR images pose a complex research task. Thanks to rapid advances in Deep Learning (DL), SR has made significant progress in recent years. Unfortunately, entry into this field is overwhelming because of the abundance of publications. It is knotty work to get an overview of the advantages and disadvantages between publications. This work unwinds some of the most representative advances in the crowded field of SR.

Existing surveys have primarily focused on classical methods [8], [9], whereas we, in contrast, concentrate exclusively on DL methods given their proven superiority. The most similar survey to this work is the one by Bashir et al. [4], which itself is a logical continuation of Wang et al. [5]. However, our work focuses on methods that have recently gained popularity: uncertainty-driven loss, advances in image quality assessment methods, new datasets, denoising diffusion probabilistic models, advances in normalization techniques, and new architecture approaches. This work also reviews pioneering work that combines neural architecture search with SR, which automatically derives designs instead of relying solely on human dexterity [10]. Finally, this work aims to give a broader overview of the field and highlight challenges as well as future trends.

Section 2 lays out definitions and introduces known metrics used by most SR publications. This section also introduces datasets and data types found in SR research. Section 3 presents commonly used regression-based learning objectives (pixel, uncertainty-driven, and content loss) as well as more sophisticated objectives like adversarial loss and denoising diffusion probabilistic models. Section 4 covers interpolation- and learning-based upsampling. Section 5 goes over the basic mechanisms of attention used in SR. Section 6 explains additional learning strategies, covering a variety of techniques that can be applied for performance improvement. It discusses curriculum learning, enhanced predictions, network fusion, multi-task learning, and normalization. Section 7 introduces SR models. It goes through different architecture types, such as simple, residual, recurrent-based, lightweight, and wavelet transform-based networks. Several graphical visualizations (also in supplementary material, (available online)) accompany the chapter, highlighting the difference between the proposed architectures. Sections 8 and 9 introduce unsupervised SR and neural architecture search combined with SR. Finally, this work summarizes and points to future directions in Sections 10 and 11.

Manuscript received 5 September 2022; revised 10 January 2023; accepted 5 February 2023. Date of publication 10 February 2023; date of current version 30 June 2023. This work was supported in part by BMBF projects ExplAINN under Grant 01IS19074, in part by EDeL under Grant 01IS19075, and in part by XAINES under Grant 01IW20005. Recommended for acceptance by A. N. Veeraraghavan MS. *(Corresponding author: Brian B. Moser.)*

Brian B. Moser and Andreas Dengel are with the German Research Center for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany, and also with the TU Kaiserslautern, 67663 Kaiserslautern, Germany (e-mail: Brian.Moser@dfki.de; andreas.dengel@dfki.de).

Federico Raue, Stanislav Frolov, Sebastian Palacio, and Jörn Hees are with the German Research Center for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany (e-mail: Federico.raue@dfki.de; Stanislav.frolov@dfki.de; sebastian.palacio@dfki.de; joern.hees@dfki.de).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3243794, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3243794

II. Setting and Terminology

This section focuses on four questions: (1) What is SR? (2) How good is a stated SR solution? (3) What SR datasets are available to test the solution? (4) How are images represented in SR?

The first question introduces the fundamental definitions as well as the terminology that is specific to SR. The second relates to evaluation metrics that assess any proposed SR solution, such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The third question is linked to numerous datasets that provide various data types, such as 8K resolution images or video sequences. Last but not least, the fourth question addresses different image representations (i.e., color spaces) that SR models can use.

A. Problem Definition: Super-Resolution

Super-Resolution (SR) refers to methods that can develop High-Resolution (HR) images from at least one Low-Resolution (LR) image [5]. The SR field divides into Single Image Super-Resolution (SISR) and Multi-Image Super-Resolution (MISR). In SISR, one LR image leads to one HR image, whereas MISR generates many HR images from many LR images. Regarding popularity, most researchers focus on SISR because its techniques are extendable to MISR.

1) Single Image Super-Resolution (SISR): The goal of Single Image Super-Resolution (SISR) is to scale up a given Low-Resolution (LR) image $\mathbf{x} \in \mathbb{R}^{\bar{w}\times\bar{h}\times c}$ to a High-Resolution (HR) image $\mathbf{y} \in \mathbb{R}^{w\times h\times c}$, with $\bar{w} \leq w$ and $\bar{h} \leq h$. Throughout this work, $N_{\mathbf{x}} = w \cdot h \cdot c$ defines the amount of pixels of an image $\mathbf{x} \in \mathbb{R}^{w\times h\times c}$ and $\Omega_{\mathbf{x}}$ the set of all valid positions in $\mathbf{x}$:

$$\Omega_{\mathbf{x}} = \{(i,j,k) \in \mathbb{N}_1^3 | i \leq h, j \leq w, k \leq c\} \tag{1}$$

Let $s \in \mathbb{N}_1$ be a scaling factor, it holds that $h = s \cdot \bar{h}$ and $w = s \cdot \bar{w}$. Furthermore, let $\mathcal{D} : \mathbb{R}^{w\times h\times c} \to \mathbb{R}^{\bar{w}\times\bar{h}\times c}$ be a degradation mapping that describes the inherent relationship between the two entities LR $(\mathbf{x})$ and HR $(\mathbf{y})$:

$$\mathbf{x} = \mathcal{D}\left(\mathbf{y}; \delta\right), \tag{2}$$

in which δ are parameters of $\mathcal{D}$ that contain, for example, the scaling factor s and other elements like blur type.

In practice, the degradation mapping is often unknown and therefore modeled, e.g., with bicubic downsampling. The underlying challenge of SISR is to perform the inverse mapping of $\mathcal{D}$. Unfortunately, this problem is ill-posed because one LR image can lead to multiple nonidentical HR images. The goal is to find a SR model $\mathcal{M} : \mathbb{R}^{\bar{w}\times\bar{h}\times c} \to \mathbb{R}^{w\times h\times c}$, s.t.:

$$\hat{\mathbf{y}} = \mathcal{M}\left(\mathbf{x}; \theta\right), \tag{3}$$

where $\hat{\mathbf{y}}$ is the predicted HR approximation of the LR image $\mathbf{x}$ and θ the parameters of $\mathcal{M}$.

For Deep Learning (DL), this translates into an optimization objective that minimizes the difference between the estimation $\hat{\mathbf{y}}$ and the ground-truth HR image $\mathbf{y}$ under a given loss function $\mathcal{L}$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right) \tag{4}$$

The setting is illustrated in the supplementary material, available online.

2) Multi-Image Super-Resolution (MISR): Multi-Image Super-Resolution (MISR) is the task of yielding one or more HR images from many LR images [11]. An example is satellite imagery [3], where many LR examples direct to a single HR prediction, a so-called many-to-one approach. An alternative is the many-to-many approach, where many LR images lead to more than one HR image. It is usually employed for video sequence enhancing [11]. The LR images are generally of the same scene, e.g., multiple satellite images of the same geographical location. Given a sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ with $T \in \mathbb{N}_1$ and $\mathbf{x}_t \in \mathbb{R}^{\bar{w}\times\bar{h}\times c}$, $0 < t \leq T$, the task is to predict $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{T'})$ with $T' \in \mathbb{N}_1$ and $\mathbf{y}_{t'} \in \mathbb{R}^{w\times h\times c}$, $0 < t' \leq T'$. The most frequent case is $T = T'$, where $\mathbf{y}_t$ is supposed to be the HR image of $\mathbf{x}_t$. Generally, MISR is an extension of the SISR setting.

B. Evaluation: Image Quality Assessment (IQA)

Many properties are associated with excellent image quality, such as sharpness, contrast, or the absence of noise. Thus, fair evaluation of SR models is challenging. This section shows different evaluation methods that fall under the umbrella term Image Quality Assessment (IQA). Broadly speaking, IQA refers to any metric based on perceptual assessments of human viewers, i.e., how realistic the image appears after applying SR methods. IQA can be subjective (e.g., human raters) or objective (e.g., formal metrics).

1) Mean Opinion Score (MOS): Digital images are ultimately meant to be viewed by human beings. Thus, the most appropriate way of assessing images is a subjective evaluation [12], [13]. One commonly used subjective IQA method is the Mean Opinion Score (MOS). Human viewers rate images with quality scores, typically 1 (bad) to 5 (good). MOS is the arithmetic mean of all ratings. Despite reliability, mobilizing human resources is time-consuming and cumbersome, especially for large datasets.

2) Peak Signal-to-Noise Ratio (PSNR): Objectively assessing quality is of indisputable importance due to the massive amount of images produced in recent years and the weaknesses of subjective measurements. One popular objective quality measurement is Peak Signal-to-Noise Ratio (PSNR). It is the ratio between the maximum possible pixel-value L (255 for 8-bit representations) and the Mean Squared Error (MSE) of reference images. Given the approximation $\hat{\mathbf{y}}$ and the ground-truth $\mathbf{y}$, PSNR is a logarithmic quantity using the decibel scale [dB]:

$$\text{PSNR}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = 10 \cdot \log_{10} \frac{L^2}{\frac{1}{N_{\mathbf{y}}} \sum_{p\in\Omega_{\mathbf{y}}} \left[\mathbf{y}_p - \hat{\mathbf{y}}_p\right]^2} \tag{5}$$

Although it is widely used as an evaluation criterion for SR models, it often leads to mediocre results in real scenarios. It focuses on pixel-level differences instead of mammalian visual perception, which is more attracted to structures [14]. Subsequently, it correlates poorly with subjectively perceived quality. Slight changes in pixels (e.g., shifting) can lead to a significantly decreased PSNR, while humans barely recognize the difference. Consequently, new metrics focus on more structural features within an image.

3) Structural Similarity Index (SSIM): The Structural Similarity Index (SSIM) depends on three relatively independent entities: luminance, contrast, and structures [14]. It is widely known and better meets the requirements of perceptual assessment [5]. SSIM estimates for an image $\mathbf{y}$ the luminance $\mu_{\mathbf{y}}$ as the mean of the intensity, while it is estimating contrast $\sigma_{\mathbf{y}}$ as its standard deviation:

$$\mu_{\mathbf{y}} = \frac{1}{N_{\mathbf{y}}} \sum_{p \in \Omega_{\mathbf{y}}} \mathbf{y}_p, \tag{6}$$

$$\sigma_{\mathbf{y}} = \frac{1}{N_{\mathbf{y}} - 1} \sum_{p \in \Omega_{\mathbf{y}}} [\mathbf{y}_p - \mu_{\mathbf{y}}]^2 \tag{7}$$

In order to compare the entities, the authors of SSIM introduced a similarity comparison function S:

$$S(x, y, c) = \frac{2 \cdot x \cdot y + c}{x^2 + y^2 + c}, \tag{8}$$

where x and y are the compared scalar variables, and $c = (k \cdot L)^2$, $0 < k \ll 1$ is a constant to avoid instability. Given a ground-truth image $\mathbf{y}$ and its approximation $\hat{\mathbf{y}}$, the comparisons on luminance ($\mathcal{C}_l$) and contrast ($\mathcal{C}_c$) are

$$\mathcal{C}_l(\mathbf{y}, \hat{\mathbf{y}}) = S(\mu_{\mathbf{y}}, \mu_{\hat{\mathbf{y}}}, c_1) \text{ and } \mathcal{C}_c(\mathbf{y}, \hat{\mathbf{y}}) = S(\sigma_{\mathbf{y}}, \sigma_{\hat{\mathbf{y}}}, c_2) \tag{9}$$

where $c_1, c_2 > 0$. The empirical co-variance

$$\sigma_{\mathbf{y},\hat{\mathbf{y}}} = \frac{1}{N_{\mathbf{y}} - 1} \sum_{p \in \Omega_{\mathbf{y}}} (\mathbf{y}_p - \mu_{\mathbf{y}}) \cdot (\hat{\mathbf{y}}_p - \mu_{\hat{\mathbf{y}}}), \tag{10}$$

determines the structure comparison ($\mathcal{C}_s$), expressed as the correlation coefficient between $\mathbf{y}$ and $\hat{\mathbf{y}}$:

$$\mathcal{C}_s(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sigma_{\mathbf{y},\hat{\mathbf{y}}} + c_3}{\sigma_{\mathbf{y}} \cdot \sigma_{\hat{\mathbf{y}}} + c_3}, \tag{11}$$

where $c_3 > 0$. Finally, the SSIM is defined as:

$$\text{SSIM}(\mathbf{y}, \hat{\mathbf{y}}) = [\mathcal{C}_l(\mathbf{y}, \hat{\mathbf{y}})]^\alpha \cdot [\mathcal{C}_c(\mathbf{y}, \hat{\mathbf{y}})]^\beta \cdot [\mathcal{C}_s(\mathbf{y}, \hat{\mathbf{y}})]^\gamma \tag{12}$$

where $\alpha > 0, \beta > 0$ and $\gamma > 0$ are adjustable control parameters for weighting relative importance of all components.

4) Learning-Based Perceptual Quality (LPQ): Lately, researchers have tried to mitigate some weak points of MOS by using DL, the so-called Learning-based Perceptual Quality (LPQ). In essence, LPQ tries to approximate a variety of subjective ratings by applying DL methods. One way is to use datasets that contain subjective scores, such as TID2013 [15], and a neural network to predict human rating scores, e.g., DeepQA [16] or NIMA [17].

A significant drawback of LPQ is the limited availability of annotated samples. One can augment a small-sized dataset by applying noise and ranking. Adding minimal noise to an image should lead to poorer quality, making the noisy image and the original counterpart pairwise discriminable. These pairs are called Quality-Discriminable Image Pairs (DIP) [18]. The DIP Inferred Quality (dipIQ) index uses RankNet [19], which is based on a pairwise learning-to-rank algorithm [20]. The authors of dipIQ show higher accuracy and robustness in variation-rich content than by training directly on IQA databases like TID2013 [15]. Another example is RankIQA [21], in which a Siamese Network [22] is used to rank image quality by using artificial distortions.

Another inventive way to calculate the similarity between two images is to use DL to extract and compare features. One well-known representative is Learned Perceptual Image Patch Similarity (LPIPS) [23], which uses L feature maps generated by an extractor φ, e.g., VGG [24]. Let H_l, W_l be the height and width of the l-th feature map, respectively, and $\alpha_l \in \mathbb{R}^{C_l}$ a scaling vector, then LPIPS is formulated as

$$\text{LPIPS}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{l=1}^{L} \frac{\sum_{h,w} \left\| \alpha_l \odot \left(\varphi^l(\hat{\mathbf{y}})_{h,w} - \varphi^l(\mathbf{y})_{h,w} \right) \right\|_2^2}{H_l W_l} \tag{13}$$

The authors showed that LPIPS aligns better with human judgments than PSNR or SSIM. However, the quality depends on the feature extractor underneath.

Another example is Deep Image Structure and Texture Similarity (DISTS) [25], which combines spatial averages (texture) with the correlations of feature maps (structure).

5) Task-Based Evaluation (TBE): Alternatively, one can focus on task-oriented features. For instance, one can measure quality differences from DL models that solve other Computer Vision (CV) tasks like image classification. Other CV tasks can also benefit from including SR as a pre-processing step. Measuring the performance on tasks, with and without SR, is yet another ingenious way to measure the quality of an SR method. HR images provide more details, which are highly desirable for CV tasks like object recognition [3]. Nevertheless, it requires extra training steps, which can be avoided by using predefined features like those presented in the following section.

6) Evaluation With Defined Features: One example is the Gradient Magnitude Similarity Deviation (GMSD) [26], which uses the pixel-wise Gradient Magnitude Similarity (GMS). Based on the variation of local quality, which arises from the diversity of local image structures, GMS calculates the gradient magnitudes with:

$$\mathbf{m}_{\mathbf{x}}(p) = \sqrt{(\nabla_h \mathbf{x})_p^2 + (\nabla_v \mathbf{x})_p^2}, p \in \Omega_{\mathbf{x}} \tag{14}$$

where $\nabla_h \mathbf{x}$ and $\nabla_v \mathbf{x}$ are the horizontal and vertical gradient images of $\mathbf{x}$, respectively. The GMS map, similar to (9), is given by

$$\text{GMS}(\mathbf{y}, \hat{\mathbf{y}})_p = S(\mathbf{m}_{\mathbf{y}}(p), \mathbf{m}_{\hat{\mathbf{y}}}(p), c) \tag{15}$$

where c is a positive constant. The final IQA score is given by the average of the GMS map:

$$\text{GMSD}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{\mathbf{y}}} \sum_{p \in \Omega_{\mathbf{y}}} \text{GMS}(\mathbf{y}, \hat{\mathbf{y}})_p \tag{16}$$

An alternative is the Feature Similarity (FSIM) Index. It also uses gradient magnitudes, but combines them with Phase Congruency (PC), a local structure measurement, as feature points [27]. The phase congruency model postulates a biologically plausible model of how human visual systems detect and identify image features. These features are perceived where the Fourier components are maximal in phase. An extension to FSIM is the Haar wavelet-based Perceptual Similarity Index (HaarPSI), which uses a Haar wavelet decomposition to assess local similarities [28]. Its authors claim that it outperforms state-of-the-art IQA methods like SSIM and FSIM in terms of execution time and higher correlations with human opinion scores. They suggest that a multi-scale complex-valued wavelet filterbank directly influences the computation of FSIM in the computation of the Fourier components. The approach is similar to FSIM but relies on 2D discrete Haar wavelet transform.

7) Multi-Scale Evaluation: In practice, SR models usually super-resolve to different scaling factors, known as Multi-Scaling (MS). Thus, evaluating metrics should address this scenario. The MS-Structural Similarity (MS-SSIM) index [29] is a direct extension of SSIM that incorporates MS. Compared to SSIM, it can be more robust to variations in viewing conditions. It adds parameters that weight the relative importance of different scales. MS-SSIM applies a low-pass filter and downsamples the filtered image by a factor of 2 for every scale level $1 \leq i \leq s_{max}$, where s_{max} is the largest scale. By doing so, it calculates the contrast and structure comparisons at each scale and the luminance comparison for the largest scale s_{max}. MS-SSIM is formulated as

$$\text{MS-SSIM}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \mathcal{C}_l\left(\mathbf{y}, \hat{\mathbf{y}}\right)^{\alpha} \cdot \prod_{i=1}^{s_{max}} \mathcal{C}_{c,i}\left(\mathbf{y}, \hat{\mathbf{y}}\right)^{\beta_i} \cdot \mathcal{C}_{s,i}\left(\mathbf{y}, \hat{\mathbf{y}}\right)^{\gamma_i}, \tag{17}$$

where $\alpha > 0$, $\beta_i > 0$ and $\gamma_i > 0$ are adjustable control parameters. Similarly, the GMSD can be extended with

$$\text{MS-GMSD}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \sqrt{\sum_{i=0}^{s_{max}} \alpha_i \cdot \left(\text{GMSD}_i\left(\mathbf{y}, \hat{\mathbf{y}}\right)\right)^2}, \tag{18}$$

where α_i are adjustable control parameters and GMSD_i is the GMSD score at i_{th} scale [30].

C. Datasets and Challenges

For SR, more extensive datasets have been made available over the past years. Various accessible datasets contain vast amounts of images of different qualities and content, as shown in Fig. 1. Some are created explicitly for SR in a supervised manner with LR-HR pairs. Moreover, one can exploit datasets that do not come with SR annotations by generating LR pairs and treating original samples as the HR version. LR samples are typically computed using bicubic interpolation, and anti-aliasing [34]. The supplementary material, available online, list commonly used datasets. Some of them were published as part of a SR challenge. Two of the most famous challenges are the New Trends in Image Restoration and Enhancement (NTIRE) challenge [35], and the Perceptual Image Restoration and Manipulation (PIRM) challenge [36].

Fig. 1. Example images from different SR datasets: Set5 [31], Set14 [1], Manga109 [32], General100 [33], BSDS100 [2], and BSDS200 [2]. The ratio of the size differences is preserved.

D. Color Spaces

Applications of SR can expand into other domains with additional modalities such as depth- or hyper-spectral SR. However relevant, we restrict the scope of this review to the more prevalent use case of color images (i.e., c=3). Data representation (i.e., color space) has played a crucial role in SR methods before DL. Nowadays, most researchers use the RGB color space [37], [38]. Nonetheless, researchers have tried to combine the advantages of other color spaces. The first DL-based SR model [39] used the first channel of the YCbCr space. It is standard to capture only the Y channel if YCbCr is employed [40] since focusing on structure instead of color is preferred. Recent SR models use the RGB color space, also because using something other than RGB color spaces is ill-defined when comparing state-of-the-art results if PSNR is used as evaluation metric [41], [42]. Exploring other color spaces for DL-based SR methods is nearly nonexistent, which presents an exciting research gap.

III. Learning Objectives

The ultimate goal of SR is to provide a model that maps an LR image to an HR image. Therefore, the question naturally arises: how to train a given SR model? The answer to this question is given in the following sections.

A. Regression-Based Objectives

Regression-based objectives attempt to model the relation between input and the desired output explicitly. The parameters of the SR model are estimated directly from the data, often by minimizing the $L1$ and $L2$ losses. While both loss functions are easily applicable, the results tend to be blurry. We also discuss one way to mitigate this shortcoming, namely by modeling uncertainty into the loss function itself.

1) Pixel Loss: The pixel loss measures the pixel-wise difference, as illustrated in the supplementary material, available online. There are two well-known pixel loss functions in literature. The first one is the Mean Absolute Error (MAE), or $L1$-loss:

$$\mathcal{L}_{\text{L1}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{1}{N_{\mathbf{y}}} \sum_{p \in \Omega_{\mathbf{y}}} \left|\mathbf{y}_p - \hat{\mathbf{y}}_p\right| \tag{19}$$

It takes the absolute differences between every pixel of both images and returns the mean value. The second well-known pixel loss function is the Mean Squared Error (MSE), or $L2$-loss. It weights high-value differences higher than low-value differences due to an additional square operation:

$$\mathcal{L}_{\text{L2}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{1}{N_{\mathbf{y}}} \sum_{p \in \Omega_{\mathbf{y}}} \left|\mathbf{y}_p - \hat{\mathbf{y}}_p\right|^2 \tag{20}$$

However, it is more common to see the $L1$-loss in literature because $L2$ is very reactive to extraordinary values: too smooth for low values and too variable for high values [38], [43]. There are variants in the literature, depending on the task and other specifications. A popular one is the Charbonnier-loss [44], which is defined by

$$\mathcal{L}_{\text{Charbonnier}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{1}{N_{\mathbf{y}}} \sum_{p \in \Omega_{\mathbf{y}}} \sqrt{\left|\mathbf{y}_p - \hat{\mathbf{y}}_p\right|^2 + \epsilon^2}, \tag{21}$$

where $0 < \epsilon \ll 1$ is a small constant such that the inner term is non-zero. Equation (19) can be seen as a special case of Charbonnier with $\epsilon = 0$.

Pixel loss functions favor a high PSNR because both formulations use pixel differences. Also, PSNR does not correlate well with subjectively perceived quality (see Section 2.2.2), which makes pixel loss functions sub-optimal. Another observation is that the resultant images tend to be blurry: sharp edges need to be modeled better. One way to combat this is by adding uncertainty.

2) Uncertainty-Driven Loss: Modeling uncertainty in DL improves the performance and robustness of deep networks [45]. Therefore, Ning et al. proposed an adaptive weighted loss for SISR [46], which aims at prioritizing texture and edge pixels that are visually more significant than pixels in smooth regions. Thus, the adaptive weighted loss treats every pixel unequally.

Inspired by insights from Variational Autoencoders (VAE) [47], they first model an estimated uncertainty. Let $\mathcal{M}$ be the SR model with parameters θ, which learns two intermediate results: $\mu_\theta(\mathbf{x})$, the mean image, and $\sigma_\theta(\mathbf{x})$, the variance image (uncertainty). Consequently, the approximated image $\hat{\mathbf{y}}$ is given by

$$\hat{\mathbf{y}} = \mathcal{M}\left(\mathbf{x}; \theta\right) = \underbrace{\mu_\theta\left(\mathbf{x}\right)}_{=\hat{\mathbf{y}}_\mu} + \epsilon \cdot \underbrace{\sigma_\theta\left(\mathbf{x}\right)}_{=\hat{\mathbf{y}}_\sigma}, \tag{22}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Most DL-based SISR methods estimate only the mean $\hat{\mathbf{y}}_\mu$. In contrast, Ning et al. proposed estimating the uncertainty $\hat{\mathbf{y}}_\sigma$ simultaneously [46].

Based on the observation that the uncertainty is sparse for SISR due to many smooth areas, Ning et al. present an Estimating Sparse Uncertainty (ESU) loss:

$$\mathcal{L}_{\text{ESU}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \exp\left(-\ln \hat{\mathbf{y}}_\sigma\right) \cdot \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_1 + 2 \cdot \ln \hat{\mathbf{y}}_\sigma \tag{23}$$

However, they observed that $\mathcal{L}_{ESU}$ lowers the performance and assumed that $\mathcal{L}_{ESU}$ is unsuitable for SISR. They also concluded that prioritizing pixels with high uncertainty is necessary to benefit from uncertainty estimation.

As a result, they proposed an adaptive weighted loss named Uncertainty-Driven Loss (UDL) and used a monotonically increasing function instead of $\exp(-\ln \hat{\mathbf{y}}_\sigma)$ in (23):

$$\mathcal{L}_{\text{UDL}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \left[\ln \hat{\mathbf{y}}_\sigma - \min\left(\ln \hat{\mathbf{y}}_\sigma\right)\right] \cdot \left\|\mathbf{y} - \hat{\mathbf{y}}_\mu\right\|_1, \tag{24}$$

where $[\ln \hat{\mathbf{y}}_\sigma - \min(\ln \hat{\mathbf{y}}_\sigma)]$ is a non-negative linear scaling function. There are two variables that need to be learned: $\hat{\mathbf{y}}_\mu$ and $\hat{\mathbf{y}}_\sigma$. Ning et al. propose to use $\mathcal{L}_{ESU}$ to learn $\hat{\mathbf{y}}_\sigma$ and then to train a new network with $\mathcal{L}_{UDL}$ to learn $\hat{\mathbf{y}}_\mu$, but with the pre-trained $\hat{\mathbf{y}}_\sigma$ fixed [46]. The idea of using two train processes is to prevent the variance image from degenerating into zeros. With this setup, they achieve better results than $L1$ and $L2$ for a wide range of network architectures [48], [49].

However, it requires a two-step training procedure, which adds extra training time. To this day, we have yet to find any techniques that circumvent such a two-step training process. The supplementary material, available online, contains additional visualizations.

3) Content Loss: Instead of using the difference between the approximated and the ground-truth image, one can transform both entities further into a more discriminant domain. Thus, the resultant loss function utilizes feature maps from an external feature extractor. Such an approach is similar to TBE (see Section 2.2.5). In more detail, the feature extractor is pre-trained on another task, i.e., image classification or segmentation. During the training of the actual SR model on the difference of feature maps, the parameters of the feature extractor remain fixed. Thus, the goal of the SR model is not to generate pixel-perfect estimations. Instead, it produces images whose features are close to the features of the target.

More formally, let φ be a Convolutional Neural Network (CNN) that extracts features like VGG [24] and let φ^l be the l-th feature map. The content loss describes the difference of feature maps generated from the HR image $\varphi^l(\mathbf{y})$ and the approximation of it, $\varphi^l(\hat{\mathbf{y}})$, and is defined as

$$\mathcal{L}_{\text{Content}}\left(\mathbf{y}, \hat{\mathbf{y}}, l\right) = \left\|\varphi^l\left(\hat{\mathbf{y}}\right) - \varphi^l\left(\mathbf{y}\right)\right\|_2 \tag{25}$$

The supplementary material, available online, provides a visualization of this approach. The motivation is to incorporate image features (content) instead of pixel-level details; a strategy that has been frequently used for Generative Adversarial Networks (GANs), e.g., SRGAN [13].

B. Generative Adversarial Networks

Since the early days of GANs [50], they have had a variety of applications in CV tasks e.g., in text-to-image synthesis [51]. The core idea is to use two distinct networks: a generator G and a discriminator D. The generator network learns to produce samples close to a given dataset and to fool the discriminator.

In the case of SR, the generator is the SR model ($G = \mathcal{M}$). The discriminator tries to distinguish between samples coming from the generator and samples from the actual dataset. The interaction between the generator and discriminator corresponds to a minimax two-player game and is optimized using the adversarial loss [12], [13]. Given the ground-truth HR image $\mathbf{y}$ of the LR image $\mathbf{x}$ and the approximated SR image $\hat{\mathbf{y}} = G_{\theta_G}(\mathbf{x})$, the loss functions based on Cross-Entropy (CE) are:

$$\mathcal{L}_G^{\text{CE}} = -\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{\text{LR}}(\mathbf{x})} \log D_{\theta_D}\left[G_{\theta_G}(\mathbf{x})\right] \tag{26}$$

$$\begin{aligned}\mathcal{L}_D^{\text{CE}} = &-\mathbb{E}_{\mathbf{y}\sim\mathbb{P}_{\text{HR}}(\mathbf{y})}\left[\log D_{\theta_D}(\mathbf{y})\right] \\ &-\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{\text{LR}}(\mathbf{x})}\left[\log\left(1 - D_{\theta_D}\left[G_{\theta_G}(\mathbf{x})\right]\right)\right]\end{aligned} \tag{27}$$

An alternative is to use the Least Square (LS) [52], [53], which yields better quality and training stability:

$$\mathcal{L}_G^{\text{LS}} = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{\text{LR}}(\mathbf{x})}\left(D_{\theta_D}\left[G_{\theta_G}(\mathbf{x})\right] - 1\right)^2 \tag{28}$$

$$\begin{aligned}\mathcal{L}_D^{\text{LS}} = &\,\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{\text{LR}}(\mathbf{x})}\left(D_{\theta_D}\left[G_{\theta_G}(\mathbf{x})\right]\right)^2 \\ &+ \mathbb{E}_{\mathbf{y}\sim\mathbb{P}_{\text{HR}}(\mathbf{y})}\left(D_{\theta_D}(\mathbf{y}) - 1\right)^2\end{aligned} \tag{29}$$

Famously known for the GAN research field, but less explored for SR [54], is the Wasserstein loss with Gradient Penalty (WGAN-GP) [55], which is the extended version of the Wasserstein loss (WGAN). Despite variations, Lucic et al. have shown that most adversarial losses can reach comparable scores with enough hyper-parameter tuning and random restarts [56]. However, this statement remains unproven for SR and requires more rigorous confirmation in the future.

A recent review by Singla et al. [57] examines GANs for SR in more detail. Generally, SR models perform better if an adversarial loss is incorporated. The primary disadvantage is that sufficient training stability is hard to reach due to GAN-specific issues like mode collapse [13]. One way to improve the training stability is to introduce regularization terms.

1) Total Variation Loss: One way to regularize GANs is to use a Total Variation (TV) denoising technique known from image processing. First introduced by Rudin et al. [58], it filters noise by reducing the TV of a given signal. The TV loss [59] measures the difference of neighboring pixels in the vertical and horizontal direction in one image. It is defined as

$$\text{TV}(\mathbf{y}) = \frac{1}{N_{\mathbf{y}}}\sum_{i,j,k}\sqrt{\underbrace{(\mathbf{y}_{i+1,j,k}-\mathbf{y}_{i,j,k})^2}_{\text{diff. first axis}} + \underbrace{(\mathbf{y}_{i,j+1,k}-\mathbf{y}_{i,j,k})^2}_{\text{diff. second axis}}} \tag{30}$$

and minimizing it results in images with smooth instead of sharp edges. This term helps to stabilize the training of GANs [13], [52]. However, it can hurt the overall image quality since sharp edges are essential.

2) Texture Loss: Texture synthesis with parametric texture models has a long history with the goal of transferring global texture onto other images [60]. Due to the advent of DL, Gatys et al. [61] proposed a style transfer method (e.g., painting style) that utilized a pre-trained neural network φ. Based on that, EnhanceNet [12] used the texture loss to enforce textural similarity. It uses the Gram Matrix to capture correlations of different channels between feature maps:

$$\mathcal{G}^l_{(c_1,c_2)}(\mathbf{y}) = \sum_{(i,j,.)\in\Omega_{\mathbf{y}}} \varphi^l(\mathbf{y})_{(i,j,c_1)} \cdot \varphi^l(\mathbf{y})_{(i,j,c_2)} \tag{31}$$

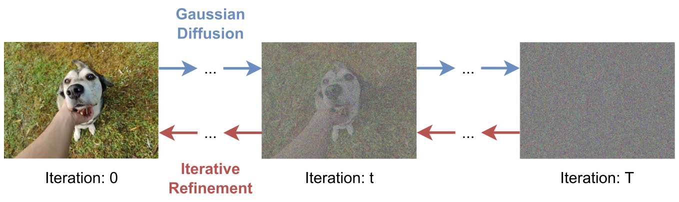

Fig. 2. Principle of DDPMs. The Gaussian diffusion process adds noise iteratively. The iterative refinement process reverts the process. The task of the SR model is to predict the noise added between two iterations. The predicted noise is then used to revert one iteration.

where φ^l is the l-th feature map. The texture loss is

$$\mathcal{L}_{\text{TEX}}(\mathbf{y}, \hat{\mathbf{y}}, l) = \left\|\mathcal{G}^l(\hat{\mathbf{y}}) - \mathcal{G}^l(\mathbf{y})\right\|_2^2 \tag{32}$$

C. Denoising Diffusion Probabilistic Models

Interestingly, DL methods are well suited for denoising Gaussian noise. Denoising Diffusion Probabilistic Models (DDPMs) [62] exploit this insight by formulating a Markov chain to alter one image into a noise distribution gradually, and the other way around. The idea of this approach is that estimating small perturbations is more tractable for neural networks than explicitly describing the whole distribution with a single, non-analytically-normalizable function.

First in SR was "Super-Resolution via Repeated Refinement" (SR3) [63]. It adds noise to the LR image until $\mathbf{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and generates a target HR image $\mathbf{y}_0$ iteratively in T refinement steps. While adding noise is straightforward, it uses a DL-based model that transforms a standard normal distribution into an empirical data distribution by reverting the noise-adding process as shown in Fig. 2.

The diffusion process q adds Gaussian noise to a LR image $\mathbf{y} = \mathbf{y}_0$ over T iterations with

$$q(\mathbf{y}_t|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_t|\sqrt{\gamma_t}\cdot\mathbf{y}_0, (1-\gamma_t)\cdot\mathbf{I}), \tag{33}$$

where $\gamma_t = \prod_{i=1}^T \alpha_i$ and $0 < \alpha_t < 1$ are hyper-parameters to determine the variance of added noise per iteration.

Consequently, the noisy image $\tilde{\mathbf{y}}$ can be expressed as

$$\tilde{\mathbf{y}} = \underbrace{\sqrt{\gamma}\cdot\mathbf{y}_0}_{\text{mean}} + \underbrace{\sqrt{1-\gamma}\cdot\epsilon}_{\text{variance}}, \quad \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}) \tag{34}$$

Given γ and $\tilde{\mathbf{y}}$, one can derive $\mathbf{y}_0$ from ϵ and vice versa by rearranging (34). Thus, a denoising model φ_θ with parameters θ has to either predict $\mathbf{y}_0$ or ϵ.

SR3 applies the denoising model to predict the noise ϵ. The model $\varphi_\theta(\mathbf{x}, \tilde{\mathbf{y}}, \gamma)$ takes as input the LR image $\mathbf{x}$, the variance of the noise γ, and the noisy target image $\tilde{\mathbf{y}}$ and is optimized by the following loss function

$$\mathcal{L}_{\text{SR3}}(\mathbf{x}, \mathbf{y}_0) = \mathbb{E}_{\epsilon,\gamma}\left\|\varphi_\theta\left(\mathbf{x}, \underbrace{\sqrt{\gamma}\cdot\mathbf{y}_0 + \sqrt{1-\gamma}\cdot\epsilon}_{=\tilde{\mathbf{y}}\text{ by Equation 34}}, \gamma\right) - \epsilon\right\|_d^d, \tag{35}$$

where $d \in \{1, 2\}$. The alternative regression target $\mathbf{y}_0$ remains open for future research.

The refinement process p is the reverse direction of the diffusion process. Given the prediction of ϵ_t by φ_θ, (34) can be reformulated to approximate $\mathbf{y}_0$:

$$\begin{aligned} \mathbf{y}_t &= \sqrt{\gamma_t} \cdot \hat{\mathbf{y}}_0 + \sqrt{1-\gamma_t} \cdot \varphi_\theta\left(\mathbf{x}, \mathbf{y}_t, \gamma_t\right) \\ \Longleftrightarrow \hat{\mathbf{y}}_0 &= \frac{1}{\sqrt{\gamma_t}} \cdot \left(\mathbf{y}_t - \sqrt{1-\gamma_t} \cdot \varphi_\theta\left(\mathbf{x}, \mathbf{y}_t, \gamma_t\right)\right) \end{aligned} \quad (36)$$

Substituting $\hat{\mathbf{y}}_0$ into the posterior distribution to parameterize the mean of $p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t, \mathbf{x})$, details shown in supplementary material, available online, yields

$$\mu_\theta\left(\mathbf{x}, \mathbf{y}_t, \gamma_t\right) = \frac{1}{\sqrt{\alpha_t}}\left[\mathbf{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} \cdot \varphi_\theta\left(\mathbf{x}, \mathbf{y}_t, \gamma_t\right)\right] \quad (37)$$

The authors of SR3 set the variance of $p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t, \mathbf{x})$ to $(1-\alpha_t)$ for the sake of simplicity. As a result, each refinement step with $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is realized as

$$\mathbf{y}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left[\mathbf{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} \cdot \varphi_\theta\left(\mathbf{x}, \mathbf{y}_t, \gamma_t\right)\right] + \sqrt{1-\alpha_t} \cdot \epsilon_t \quad (38)$$

And indeed, SR3 generates sharp images with more details than its regression-based counterparts on natural and face images. The authors also strengthened the results by positively testing the generated images with 50 human subjects. Thus, the outcome of SR3 is a promising direction for future investigations. Nevertheless, SR3 seems prone to bias, e.g., applied to a facial dataset; the images result in too smooth skin texture by dropping moles, pimples, and piercings. More extensive ablation studies could help in establishing the practicality for real-world SR applications.

IV. Upsampling

A critical aspect is how to increase the spatial size of a given feature map. This section gives an overview of upsampling methods, either interpolation-based (nearest-neighbor, bilinear, and bicubic interpolation) or learning-based (transposed convolution, sub-pixel layer, and meta-upscale). Various visualizations can be found in the supplementary material, available online.

A. Interpolation-Based Upsampling

Many DL-based SR models use image interpolation methods because of their simplicity. The most known methods are nearest-neighbor, bilinear, and bicubic interpolation.

Nearest-neighbor interpolation is the most straightforward algorithm because the interpolated value is based on its nearest pixel values. Nearest-neighbor is swift since no calculations are needed, which is why it is favorable for SR models [12]. However, there are no interpolated transitions, which results in blocky artifacts.

In contrast, bilinear interpolation bypasses blocky artifacts by producing smoother transitions with linear interpolation on both axes. It needs a receptive field of 2×2, making it relatively fast and easily applicable [64].

However, SR methods typically do not use it because bicubic interpolation delivers much smoother results. But it demands much more computation time due to a receptive field size of 4×4 and makes it the slowest approach among all three methods. Nonetheless, it is generally used by SR models if an interpolation-based upsampling is applied [39], [44], [65] since the overall time consumption is unessential for GPU-based models, which are common in SR research.

B. Learning-Based Upsampling

Learning-based upsampling introduces modules that upsample a given feature map within a learnable setup. The most standard learning-based upsampling methods are the transposed convolution and sub-pixel layer. Promising alternatives are meta-upscaling, decomposed upsampling, attention-based upsampling, and upsampling via Look-Up Tables, which are the last discussed methods in this section.

1) Transposed Convolution: Transposed convolution expands the spatial size of a given feature map and subsequently applies a convolution operation. In general, the expansion is realized by adding zeros between given values. However, some approaches differ by first applying nearest-neighbor interpolation and then applying zero-padding, e.g., FSRCNN [33]. The receptive field of the transposed convolution layer can be arbitrary (often set to 3×3). The added values depend on the kernel size of the subsequent convolution operation. This procedure is widely known in literature and is also called deconvolution layer, although it does not apply deconvolution [39], [40], [43]. In practice, transposed convolution layers tend to produce crosshatch artifacts due to zero-padding (further discussed in the supplementary material, available online). Also, the upsampled feature values are fixed and redundant. The sub-pixel layer was proposed to circumvent this problem.

2) Sub-Pixel Layer: Introduced with ESPCN [66], it uses a convolution layer to extract a deep feature map and rearranges it to return an upsampled output. Thus, the expansion is carried out in the channel dimension, which can be more efficient for smaller kernel sizes than transposed convolution. However, assume zero values are used instead of nearest-neighbor interpolation in the transposed convolution. In that case, a transposed convolution can be simplified to a sub-pixel layer [67]. Given a scaling factor s and input channel size c, it produces a feature map with $s^2 \cdot c$ channels. Next, a convolution operation with zero padding is applied, such that the spatial size of the input and the resulting feature map remains the same. Finally, it reshapes the feature map to produce a spatially upsampled output. The receptive field of the convolutional layer can be arbitrary but is often 1×1 or 3×3. This procedure is widely known in the literature as pixel shuffle layer [38], [53], [63]. In practice, the sub-pixel layer produces repeating artifacts that can be difficult to unlearn for deeper layers, which is also further explained in the supplementary material, available online.

3) Decomposed Upsampling: An extension to the above approaches is decomposed transposed convolution [68]. Using 1D convolutions instead of 2D convolutions reduces the number of operations and parameters for the component k^2 to $2 \cdot k$. Note that the applied decomposition is not exclusively bound to transposed convolution and can also be used in sub-pixel layers.

4) Attention-Based Upsampling: Another alternative to transposed convolution is attention-based upsampling [69]. It follows the definition of attention-based convolution (or scaled dot product attention) and replaces the 1x1 convolutions with upsampling methods. In more detail, it replaces the convolution for the query matrix with bilinear interpolation and the convolution for the key and value matrix with zero-padding upsampling similar to transposed convolution. It needs fewer parameters than transposed convolution but a slightly higher number of operations. Also, it trains considerably slower: a shared issue across attention mechanisms.

5) Upsampling With Look-Up Tables: A recently proposed alternative is to use a Look-Up Table (LUT) to upsample [70]. Before generating the LUT, a small-scale SR model is trained to upscale small patches of a LR image to target HR patches. Subsequently, the LUT is created by saving the results of the trained SR model applied on a uniformly distributed input space. It reduces the upsampling runtime to the time necessary for memory access while achieving better quality than bicubic interpolation. On the other hand, it requires additional training to create the LUT. Also, the size of the small patches is crucial since the size of the LUT increases exponentially. However, it is an exciting approach that is worth further exploration.

6) Flexible Upsampling: Most existing SR methods only consider fixed, integer scale factors $s \in \mathbb{N}$. However, most real-world scenarios require flexible zooming, which demands SR methods that can be applied with arbitrary scale factors $s \in \mathbb{R}^+$. It raises a significant problem for models that use those learning-based upsampling methods: one has to train and save various models for different scales, which limits the use of such methods for real-world scenarios [37], [66]. In order to overcome this limitation, a meta-upscale module was proposed [41]. It predicts a set of filters for each position in a feature map that is later applied to a location in a lower-resolution feature map. A visualization of the upscale module and the aforementioned steps can be found in the supplementary material, available online. This approach is exciting for tasks that require arbitrary magnification levels, e.g., zooming. Nevertheless, its downsides unfold if huge scaling factors are required because it has to predict the filter weights $W(i, j)$ independently for each position. There are only a few works [71] that involve this approach in the literature. Besides existing SR methods, one can treat each image as a continuous function and generate patches at a fixed pixel resolution around a center like AnyRes-GAN (2022) [72], which could be an interesting path to follow for future SR research.

V. Attention Mechanisms for SR

Attention revolutionized Natural Language Processing (NLP) and plays an essential role for DL in several applications [73]. This section presents how SR methods employ attention. There are two categories: Channel and spatial attention, which can be applied simultaneously.

A. Channel-Attention

Feature maps generated by CNNs are not equally important. Therefore, essential channels should be weighted higher than counterpart channels, which is the goal of channel attention. It focuses on "which" (channels) carry crucial details. Hu et al. [74] introduced the channel-attention mechanism for CV as an add-on module for any CNN architecture. Fig. 3 illustrates the principle. Upon this, the Residual Channel Attention Network (RCAN) [37] first applied the channel-attention to SR. They used a second-order feature metric to reduce the spatial size and applied two fully connected layers to extract the importance weighting. Because of its simple add-on installation, it is still in use for current research [75]. One interesting future direction of channel-attention would be to combine it with pruning in order to not only amplify quality but also to save computation time.

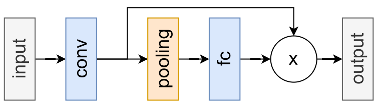

Fig. 3. Channel-attention mechanism [74]. It reduces a feature map in the spatial dimensions and extracts weighting values by using several FC layers that are element-wise multiplied to the initial feature map.

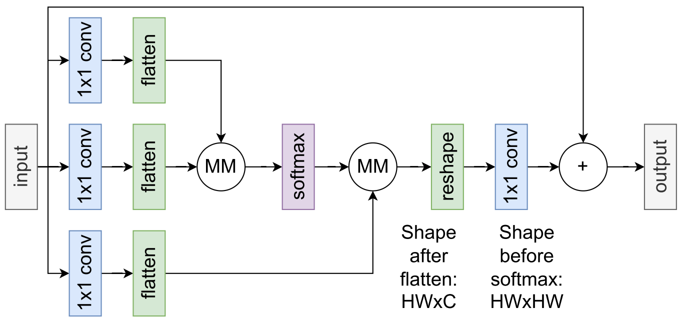

Fig. 4. Spatial-attention mechanism [76]. It extracts informations by inspecting the relationship between two positions (first matrix multiplication) and returns the importance of each position as a feature map (second matrix multiplication). MM denotes the matrix multiplication.

B. Spatial-Attention

In contrast to channel attention, spatial attention focuses on "where" the input feature maps carry important details, which requires extracting global information from the input. Therefore, Wang et al. [76] proposed non-local attention, similar to self-attention for NLP, following the idea of Query-Key-Value-paradigm known from transformers [73].

The principle is shown in Fig. 4. Intuitively, the resulting feature map contains each position's weighted sum of features. It captures spatially separated information by a long range in the feature map. In SR, the Residual Non-local Attention Network (RNAN) [77] was the first architecture using non-local spatial attention to improve performance.

A known architecture is SwinIR [78], which reaches state-of-the-art performance and follows the idea of transformers (i.e., multi-head self-attention [73]) more strictly by applying a sequence of multiple swin transformer layers. SwinFIR [79] develops this idea further with fast fourier convolutions. Since transformers originate from NLP, their natural application is restricted to a sequence of patches, and therefore, it requires a patch-wise formulation of an image. This is a significant drawback of transformers applied to SR. SwinIR diminishes the problem by shifting between attention blocks to transfer knowledge between patches.

Exploring its full potential is an exciting topic for the future [80], [81]. However, one major drawback is the difficulty of measuring the importance at a global spatial scale since it often requires vector multiplications (along rows and columns), which is slow for large-scale images. Bottleneck tricks and patch-wise applications are used to circumvent this problem but are also sub-optimal for importance weighting. Future research should further optimize computation speed without forfeiting global information.

C. *Mixed Attention*

Since both attention types can be applied easily, merging them into one framework is natural. Thus, the model focuses on “which” (channel) is essential and “where” (spatially) to extract the most valuable features. This combines the benefits of both approaches and introduces an exciting field of research, especially in SR [82], [83]. One potential future direction would be to introduce attention mechanisms incorporating both concerns in one module.

A well-known architecture for mixed attention is the Holistic Attention Network (HAN) [82], which uses hierarchical features (from residual blocks) and obtains dependencies between features of different depths while allocating attention weights to them. Thus, it weighs the importance of depth layers in the residual blocks instead of spatial locations within one feature map. Finally, the network uses the layer attention weighted feature map and combines it via addition with channel attention before the final prediction.

VI. Additional Learning Strategies

Additional learning strategies are blueprints that can be used in addition to regular training. This section presents the most common methods used in SR, which can immensely impact an SR model’s overall performance.

A. *Curriculum Learning*

Curriculum learning follows the idea of training a model under easy conditions and gradually involving more complexity [84], i.e., additional scaling sizes. For instance, a model trains with a scale of two and gradually higher scaling factors, e.g., ProSR [53]. ADRSR [85] concatenates all previously computed HR outputs. CARN [86] updates the HR image in sequential order, where additional layers extend the network for four times upsampling, which uses previously learned layers, but ignores previously computed HR images [53]. Another way of using curriculum learning is gradually increasing the noise in LR images, e.g., in SRFBN [64]. Curriculum learning can shorten the training time by reducing the difficulty of large scaling factors.

B. *Enhanced Predictions*

Instead of enhancing simple input-output pairs, one can use data augmentation techniques like rotation and flipping for final prediction. More specifically, create a set of images via data augmentation of one image, e.g., rotation. Next, let the SR model reconstruct images of the set. Finally, inverse the data augmentation via transformations and derive a final prediction, i.e., the mean [40], or the median [87].

C. *Learned Degradation*

So far, this work assumed learning processes from LR to HR. The Content Adaptive Resampler (CAR) [7] introduced a resampler for downscaling. It predicts kernels to produce downscaled images according to its HR input. Next, a SR model takes the LR image and predicts the SR image. Thus, it simultaneously learns the degradation mapping and upsampling task. The authors demonstrate that the CAR framework trained jointly with SR networks improves state-of-the-art SR performance furthermore.

D. *Network Fusion*

Another way of improving performance is to apply multiple SR models. Network fusion uses the output of all additional SR models and applies a fusion layer to the outputs. Finally, it predicts the SR image used for the learning objective. For instance, the Context-wise Network Fusion (CNF) [88] uses this approach with SRCNN, a small SR model introduced in the next chapter. This method adds a considerable amount of computation and memory, depending on the complexity of the SR models. However, it achieves a performance boost and is recommended if computational resources are sufficiently available. A potential future direction would be to use network fusion to determine critical computation paths and to merge multiple networks into one by pruning or dropout mechanisms.

E. *Multi-Task Learning*

Multi-task learning is an exciting research area for SR with the core idea to train a given model to perform various tasks [52], [85]. E.g., one can assign a label to each image and use multiple datasets for training. Next, a SR model can learn to reconstruct the SR image and predict its category (e.g., natural or manga image) [89]. Another idea is to use datasets that have access to salient image boundaries or image segmentation and use features extracted from the model to predict these [90]. Research on self-learning is an interesting starting point to combine ideas with SR.

F. *Normalization Techniques*

A slight change in the input distribution is a cause of many issues because layers need to continuously adapt to new distributions, which is known as covariate shift and can be alleviated with BatchNorm [91]. While BatchNorm helps in tasks like image classification, it is frequently used for SR [13], [52], [92]. However, BatchNorm does not apply well in SR because it

TABLE I
COMPARISON (PSNR/SSIM) OF SR APPROACHES DISCUSSED IN THIS WORK: SIMPLE, RESIDUAL, RECURRENT, ATTENTION-BASED, LIGHTWEIGHT AND WAVELET NETWORKS, AS WELL AS NAS DERIVED NETWORKS AND UNSUPERVISED TRAINED MODELS. THE ORDERING REFLECTS ROUGHLY THE PERFORMANCE RANKING. HOWEVER, SOME COMPARISONS ARE NOT FAIR DUE TO DIFFERENT SETTINGS, E.G., NAS VERSUS UNSUPERVISED RESULTS. ALSO, SOME PUBLICATIONS DO NOT REPORT RESULTS ON THE COMMONLY USED DATASETS. THEREFORE, THEY ARE NOT LISTED (E.G., WESPE [128], DSN [133], CINCGAN [52])

Model	Set5 [31]			Set14 [1]			BSDS100 [2]		
	x2	x3	x4	x2	x3	x4	x2	x3	x4
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
Bicubic	33.66/0.9542	30.39/0.8682	28.42/0.8104	30.24/0.8688	27.55/0.7742	26.00/0.7027	29.56/0.8431	27.21/0.7385	25.96/0.6675
DIP [131]	n.a.	n.a.	28.90/n.a.	n.a.	n.a.	27.00/n.a.	n.a.	n.a.	n.a.
NAS-DIP [144]	35.90/n.a.	n.a.	31.09/n.a.	31.89/n.a.	n.a.	28.37/n.a.	n.a.	n.a.	n.a.
SRCNN [39]	36.66/0.9542	32.75/0.9090	30.48/0.8628	32.42/0.9063	29.28/0.8209	27.49/0.7503	31.36/0.8879	28.41/0.7863	26.90/0.7101
FSRCNN [33]	37.00/0.9558	33.16/0.9140	30.71/0.8657	32.63/0.9088	29.43/0.8242	27.59/0.7535	31.51/0.8910	n.a.	26.97/0.7140
ESPCN [66]	n.a.	33.13/n.a.	30.90/n.a.	n.a.	29.49/n.a.	27.73/n.a.	n.a.	n.a.	n.a.
MZSR [135]	37.25/0.9567	n.a.	n.a.	n.a.	n.a.	n.a.	31.64/0.8928	n.a.	n.a.
ZSSR [87]	37.37/0.9570	33.42/0.9188	31.13/0.8796	33.00/0.9108	29.80/0.8304	28.01/0.7651	31.65/0.8920	28.67/0.7945	27.12/0.7211
LapSRN [43]	37.52/0.9591	33.81/0.9220	31.54/0.8852	32.99/0.9124	29.79/0.8325	28.09/0.7700	31.80/0.8952	28.82/0.7980	27.32/0.7275
VDSR [98]	37.53/0.9587	33.66/0.9213	31.35/0.8838	33.03/0.9124	29.77/0.8314	28.01/0.7674	31.90/0.8960	28.82/0.7976	27.29/0.7251
DWSR [120]	37.43/0.9568	33.82/0.9215	31.39/0.8833	33.07/0.9106	29.83/0.8308	28.04/0.7669	31.80/0.8940	n.a.	27.25/0.7240
DRCN [105]	37.63/0.9588	33.82/0.9226	31.53/0.8854	33.04/0.9118	29.76/0.8311	28.02/0.7670	31.85/0.8942	28.80/0.7963	28.23/0.7233
MoreMNAS [10]	37.57/0.9584	n.a.	n.a.	33.25/0.9142	n.a.	n.a.	31.94/0.8966	n.a.	n.a.
RED [101]	37.66/0.9599	33.82/0.9230	31.51/0.8869	32.94/0.9144	29.61/0.8341	27.86/0.7718	31.99/0.8974	28.93/0.7994	27.40/0.7290
DSRN [97]	37.66/0.9590	33.88/0.9220	31.40/0.8830	33.15/0.9130	30.26/0.8370	28.07/0.7700	32.10/0.8970	28.81/0.7970	27.25/0.7240
DRRN [65]	37.74/0.9591	34.03/0.9244	31.68/0.8888	33.23/0.9136	29.96/0.8349	28.21/0.7720	32.05/0.8973	28.95/0.8004	27.38/0.7284
CARN-M [112]	37.53/0.9583	33.99/0.9236	31.92/0.8903	33.26/0.9141	30.08/0.8367	28.42/0.7762	31.92/0.8960	28.91/0.8000	27.44/0.7304
MemNet [92]	37.78/0.9597	34.09/0.9248	31.74/0.8893	33.28/0.9142	30.00/0.8350	28.26/0.7723	32.08/0.8978	28.96/0.8001	27.40/0.7281
IDN [43]	37.83/0.9600	34.11/0.9253	31.82/0.8903	33.30/0.9148	29.99/0.8354	28.25/0.7730	32.08/0.8985	28.95/0.8013	27.41/0.7297
FALSR [140]	37.82/0.9595	n.a.	n.a.	33.55/0.9168	n.a.	n.a.	32.12/0.8987	n.a.	n.a.
MWCNN [123]	37.91/0.9600	34.17/0.9271	32.12/0.8941	33.70/0.9182	30.16/0.8414	28.41/0.7816	32.23/0.8999	29.12/0.8060	27.62/0.7355
NLRN [109]	38.00/0.9603	34.27/0.9266	31.92/0.8916	33.46/0.9159	30.16/0.8374	28.36/0.7745	32.19/0.8992	29.06/0.8026	27.48/0.7306
CARN [112]	37.76/0.9590	34.29/0.9255	32.13/0.8937	33.52/0.9166	30.29/0.8407	28.60/0.7806	32.09/0.8978	29.06/0.8034	27.58/0.7349
IMDN [115]	38.00/0.9605	34.36/0.9270	32.21/0.8948	33.63/0.9177	30.32/0.8417	28.58/0.7811	32.19/0.8996	29.09/0.8046	27.56/0.7353
RFDN [114]	38.05/0.9606	34.41/0.9273	32.24/0.8952	33.68/0.9184	30.34/0.8420	28.61/0.7819	32.16/0.8994	29.09/0.8050	27.57/0.7360
RFDN-L [114]	38.08/0.9606	34.47/0.9280	32.28/0.8957	33.67/0.9190	30.35/0.8421	28.61/0.7818	32.18/0.8996	29.11/0.8053	27.58/0.7363
DeCoNAS [141]	37.96/0.9594	n.a.	n.a.	33.63/0.9175	n.a.	n.a.	32.15/0.8986	n.a.	n.a.
HNAS [143]	38.11/0.9640	n.a.	n.a.	33.60/0.9200	n.a.	n.a.	32.17/0.9020	n.a.	n.a.
ESRN [137]	38.04/0.9607	34.46/0.9281	32.26/0.8957	33.71/0.9185	30.43/0.8439	28.63/0.7818	32.23/0.9005	29.15/0.8072	27.62/0.7378
SRFBN [64]	38.11/0.9609	34.70/0.9292	32.47/0.8983	33.82/0.9196	30.51/0.8461	28.81/0.7868	32.29/0.9010	29.24/0.8084	27.72/0.7409
SRFBN+ [64]	38.18/0.9611	34.77/0.9297	32.56/0.8992	33.90/0.9203	30.61/0.8473	28.87/0.7881	32.34/0.9015	29.29/0.8093	27.77/0.7419
MDSR [38]	38.17/0.9605	34.77/0.9288	32.60/0.8982	33.92/0.9203	30.53/0.8465	28.82/0.7876	32.34/0.9014	29.30/0.8101	27.78/0.7425
EDSR [38]	38.20/0.9606	34.76/0.9290	32.62/0.8984	34.02/0.9204	30.66/0.8481	28.94/0.7901	32.37/0.9018	29.32/0.8104	27.79/0.7437
WRAN [125]	38.32/0.9630	34.79/0.9310	32.36/0.8990	34.21/0.9220	30.71/0.8520	28.60/0.7860	32.57/0.9070	29.36/0.8190	27.71/0.7420
DRLN [6]	38.27/0.9616	34.78/0.9303	32.63/0.9002	34.28/0.9231	30.73/0.8488	28.94/0.7900	32.44/0.9028	29.36/0.8117	27.83/0.7444
HAN [82]	38.33/0.9617	34.85/0.9305	32.75/0.9016	34.24/0.9224	30.77/0.8495	28.99/0.7907	32.45/0.9030	29.39/0.8120	27.85/0.7454
DRLN+ [6]	38.34/0.9619	34.86/0.9307	32.74/0.9013	34.43/0.9247	30.80/0.8498	29.02/0.7914	32.47/0.9032	29.40/0.8125	27.87/0.7453
SwinIR [149]	38.46/0.9624	35.04/0.9322	32.93/0.9043	33.07/0.9106	31.00/0.8542	29.15/0.7958	31.80/0.8940	29.49/0.8150	27.95/0.7494
WIDN [121]	39.60/0.9830	34.48/0.9430	32.85/0.9290	34.44/0.9800	30.95/0.9310	29.75/0.9090	33.52/0.9790	29.99/0.9280	28.10/0.9100
CAR (EDSR) [7]	38.94/0.9658	n.a.	33.88/0.9174	35.61/0.9404	n.a.	30.31/0.8382	33.83/0.9262	n.a.	29.15/0.8001

removes network range flexibility by normalizing features [38]. The performance of a SR model can be increased substantially without BatchNorm [93].

Recently, the authors of Adaptive Deviation Modulator (AdaDM) [94] studied this phenomenon and were able to show that the standard deviation of residual features shrinks a lot after normalization layers (including BatchNorm, layer, instance, and group normalization), which causes the performance degradation in SR. The standard deviation resembles the amount of variation of pixel values, which primarily affects edges in images. They proposed a module within a feature extraction block to address this problem and amplify the pixel deviation of normalized features. It calculates a modulated output $\widetilde{\mathbf{y}}$ by

$$\widetilde{\mathbf{y}} = \mathbf{y} \cdot \mathrm{e}^{\varphi[\log(\sigma[\mathbf{x}])]}, \tag{39}$$

where $\mathbf{x}$ is the residually propagated input, $\sigma[\mathbf{x}]$ its standard deviation along all three axes, and φ a learnable feature extraction module. The logarithmic scaling guarantees stable training. As a result, it is possible to apply normalization layers to state-of-the-art SR models and to achieve substantial performance improvements. One limitation is the additional GPU memory during training, roughly two times the size. Therefore, lightweight normalization techniques similar to AdaDM, but with less GPU memory consumption are of high interest for future research.

VII. SR MODELS

This section presents the most technical part: How to construct an SR model? In the beginning, we introduce different frameworks that all architectures need to apply, which is the location of the upsampling itself. After that, it chronologically examines different architecture types. Additional visualizations can be found in the supplementary material, available online, and an overview benchmark in Table I.

A. Upsampling Location

Besides the choice of the upsampling method, one crucial decision in designing SR models is where to place the upsampling in the architecture. There are four variants explored in SR literature [4]: Pre-, post-, progressive, and iterative up-and-down upsampling. They are shown in Fig. 5 and discussed next. An

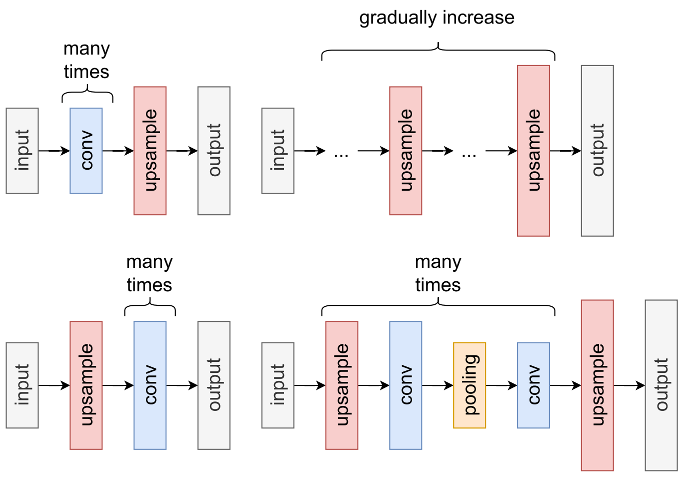

Fig. 5. Visualization of different upsampling locations within a neural network: (upper left) post-upsampling, (bottom left) pre-upsampling, (upper right) progressive upsampling, and (bottom right) iterative up-and-down upsampling.

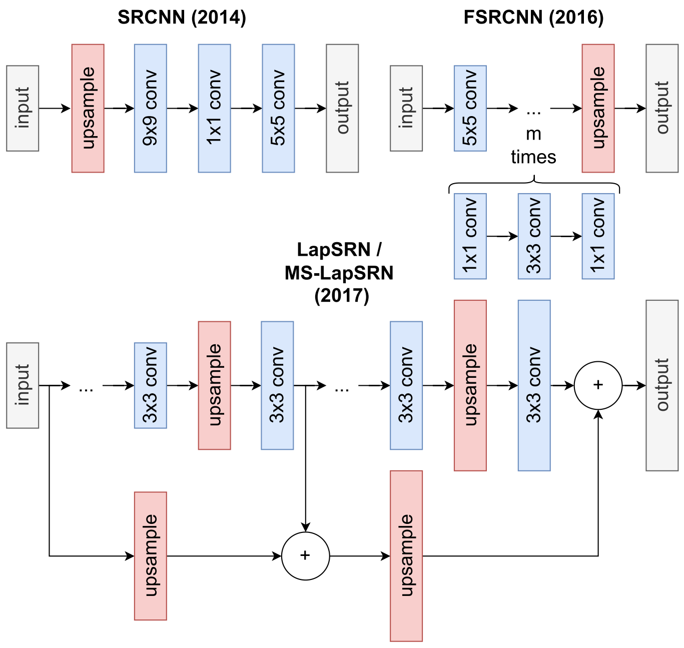

Fig. 6. Architecture designs of SRCNN [39], FSRCNN [33], ESPCN [66] and LapSRN/MS-LapSRN [44]. They are grouped together in this work as simple network designs, because they use only convolutional operations and upsampling methods.

additional overview table can be found in the supplementary material, available online.

Pre-Upsampling. In a pre-upsampling framework, upsampling is applied in the beginning. Hereafter, it uses a composition of convolution layers to extract features. Since the feature maps match the spatial size of the HR image early, the convolutional layers focus on refining the upsampled input. Thus, it eases learning difficulty but introduces blur and noise amplification in the upsampled image, which impacts the final result. Also, models trained with pre-upsampling have less performance or application issues than other frameworks since they use DL to refine independently of the scaling in the beginning. However, it also comes with high memory and computational costs compared to other frameworks like post-upsampling because of the larger input for feature extraction.

Post-Upsampling. Post-upsampling lowers memory and computation costs by upsampling at the end, which results in feature extractions in lower dimensional space. Due to this advantage, this framework decreases the complexity of SR models, e.g., FSRCNN [33]. However, the computational burden advances for multi-scaling, where approximations at different scales are required. A step-wise upsampling during the feed-forward process was proposed to alleviate this issue, the so-called progressive upsampling.

Progressive Upsampling. Unlike pre- and post-upsampling, progressive upsampling gradually increases the feature map size within the architecture. Cascaded CNN-based modules typically enhance to a single scaling factor. Its output is the input for the next module, typically built similarly. Thus, it segregates the upscaling problem into small tasks, which is a perfect fit for multi-scale SR tasks. Also, it reduces the learning difficulty for convolutional layers. However, the benefit is limited to the highest set scale, and higher scaling factors also require deeper architectures.

Iterative Up-and-Down Upsampling. Learning the mapping from HR to LR helps to understand the relation from LR to HR [95]. Iterative up-and-down upsampling uses this insight by also downsampling within the architecture. It is found mainly within recurrent-based network designs [96], [97] and demonstrated significant improvement compared to the previous upsampling techniques. However, the proper use requires further exploration. It can be applied with other frameworks, an interesting avenue for future research.

B. *Simple Networks*

Simple networks are architectures that mainly apply a chain of convolutions. They are easy to understand and typically use a bare minimum of computational resources due to their size. Most of these architectures can be found in the early days of DL-based SR because their performances are below state-of-the-art. Also, the “the deeper, the better” paradigm of DL does not apply well for simple networks because of vanishing/exploding gradients [98]. Fig. 6 shows the different simple network designs. The first CNN introduced for SR datasets was SRCNN (2014) by Dong et al. [39]. It uses bicubic pre-upsampling to match the ground-truth spatial size (see Section 7.1). Subsequently, it consists of three convolution layers, which followed a popular strategy in image restoration: patch extraction, non-linear mapping, and reconstruction. The authors of SRCNN claimed that applying more layers hurts the performance, which contradicts the DL paradigm “the deeper, the better” [99]. As seen in the following, this observation was false and required more advanced building blocks to work correctly, e.g., residual connections like in VDSR [98].

In their follow-up paper, the authors explored various ways to speed up SRCNN, resulting in FSRCNN (2016) [33] that utilized three major tricks: First, they reduced the kernel size of

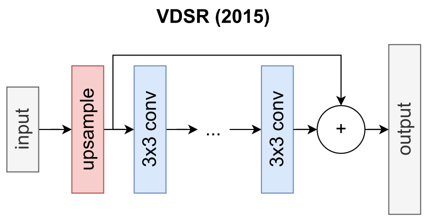

Fig. 7. Architecture design of VDSR [98]. It applies a deep convolutional network as feature extractor but also uses a residual from the input to the final prediction. Therefore, the feature extractor can concentrate to add high-frequency details onto the interpolated image.

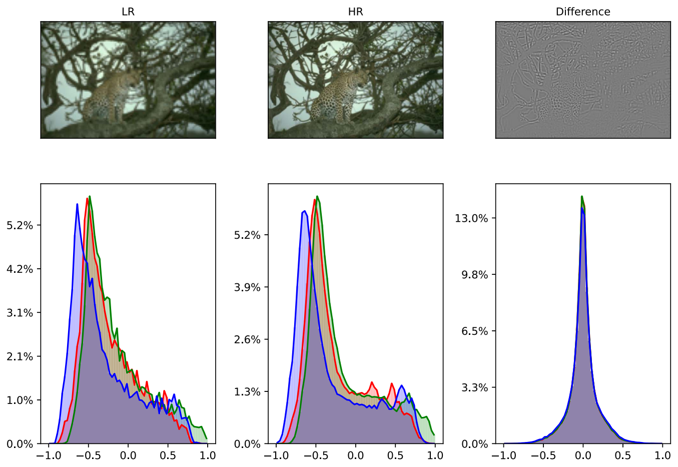

Fig. 8. Pixel value distribution of an example image from BSDS100 [2]. It can be observed that the distribution of the LR and HR image are skewed. In contrary, the distribution of the difference between LR and HR looks normally distributed, which is easier learned by an SR model.

convolution layers. Second, they used a 1x1 convolution layer to enhance and reduce the channel dimension before and after a feature processing with 3x3 convolutions. Third, they applied post-upsampling with transposed convolution, which is the main reason for the speed up (see Section 7.1). Surprisingly, they outperformed SRCNN while obtaining faster computation.

A year later, LapSRN (2017) [44] was proposed with a key contribution of a Laplacian pyramid structure [100] that enables progressive upsampling (see Section 7.1). It takes coarse-resolution feature maps as input and predicts high-frequency residuals that progressively refine the SR reconstruction at each pyramid level. To this end, predicting multi-scale images in one feed-forward pass is feasible, thereby facilitating resource-aware applications.

Simple network architecture designs are found primarily in the early days of DL-based SR because of their limited capacity to learn complex structures due to their size. Recently, researchers focused on networks with more depth, either with residual networks or synthetic depth with recurrent-based networks. The following sections introduce both possibilities.

C. Residual Networks

Residual networks use skip connections to jump over layers. The primary reason behind adding skip connections is twofold: To avoid vanishing gradients and mitigate the accuracy saturation problem [99]. For SR, introducing skip connections unlocked the world of deep-constructed models. The main advantage is that deep architectures substitute convolutions with large receptive fields, which are crucial for capturing important features. The authors of SRCNN stated that the "the deeper, the better" paradigm does not hold to SR. In contrast, Kim et al. refuted this statement with VDSR (2015) [98] and showed that very deep networks could significantly improve SR, visualized in Fig. 7.

They used two insights from other DL approaches: First, they applied a famous architecture VGG-19 [24] as a feature extraction block. Second, they used a residual connection from the interpolation layer to the last layer. As a result, the VGG-19 feature extraction block adds high-frequency details to the interpolation, resulting in a target distribution that is normal and reduces the learning difficulty immensely, as shown in Fig. 8. Also, it diminishes vanishing/exploding gradients due to the sparse representation of the high-frequency added to the interpolation. This merit emitted a trend for following residual networks that enhance the number of residuals used.

One example is RED-Net (2016) [101], which adapts the U-Net [102] architecture to SR. It incorporates a downsampling encoder and an upsampling decoder network, which downsamples the given output to extract features and then upsamples the feature maps to a target spatial size. During this process, RED-Net extends the upsample operation with residual information acquired throughout the downsample procedure, which reduces vanishing gradients. As a result, it outperforms SRCNN for several scaling factors.

Another example adapted ResNet [103], which consists of multiple residual units, and DenseNet [104], which sends residual information to all later appearing convolution operations, in the publication of SRGAN (2016) [13]. Moreover, the authors compared these architectures applied with pixel and adversarial loss (see Section 3). SRResNet consists of multiple stacked residual units that allow high-level feature extractions to access low-level feature information through numerous summation operations. Therefore, it eases optimization by providing an easy back-propagation path to early layers. Like SRResNet, SRDenseNet [40] applies dense residual blocks, which utilize even more residual connections to allow direct paths to earlier layers. In comparison, SRResNet outperforms SRCNN, DRCN, and ESPCN by a large margin. An extension to SRDenseNet is the Residual Dense Network [42], proposed in 2018, which incorporates an additional residual connection upon the dense block.

The Densely Residual Laplacian Network (DRLN) [6] is an extension of SRDenseNet and a post-upsampling, channel attention-based residual network and achieves competitive results regarding state-of-the-art. Each dense block is followed by a module based on Laplacian pyramid attention, which learns inter and intra-level dependencies between feature maps. It weights sub-band features progressively in each DRLM,

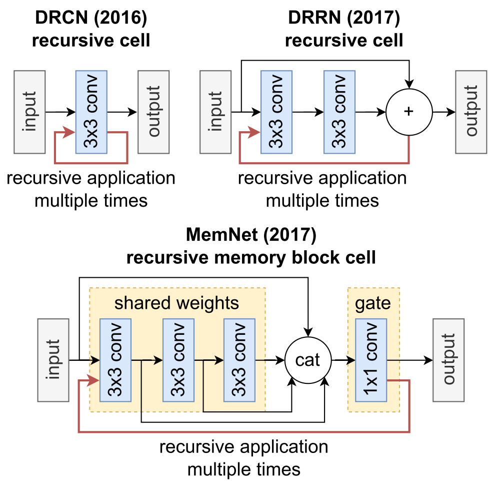

Fig. 9. Recurrent-based architecture designs of DRCN [105] (upper left), DRRN [65] (upper right), and MemNet [92] (bottom). The DRCN applies the same convolutional layer multiple times. The DRNN extends this idea to recursive application of a residual block. The MemNet introduces a gate unit to distill most important features from all intermediate convolution results in a recursive manner.

similar to HAN (concatenating various feature maps of different depths).

Another variant of residual blocks is the Information Distillation Network (IDN) [43]. It employs residual connections to accumulate a portion of a feature map to later layers. Given six convolution layers, it divides the feature map into two parts in the middle. Then, one part is processed further by the last three layers and added to the concatenation of the input and the other part. In brief, networks that utilize residual connections are state-of-the-art. Their ability to efficiently propagate information helps fight vanishing/exploding gradients, resulting in excellent performances. Sometimes, residual block usage is combined with other architectures, such as recurrent-based networks.

D. *Recurrent-Based Networks*

Artificial depth can be accomplished with recurrence, where the receptive field, crucial for capturing important information, is enlarged by repeating the same operation. Also, recurrence reduces the number of parameters, which helps to combat overfitting and memory consumption for small devices. It is achieved by applying a convolution layer multiple times without introducing new parameters.

The first recurrent-based network for SR was introduced with DRCN (2015) by Kim et al. [105]. It uses the same convolution layer up to 16 times, and a subsequent reconstruction layer considers all recursive outputs for final estimation. However, they observed that their deeply-recursive network is hard to train but eased it with skip connections and recursive supervision, essentially auxiliary training. Fig. 9 visualizes DRCN.

Combining the core ideas of DRCN [105] and VDSR [98], DRRN (2017) [65] utilizes several stacked residual units in a recursive block structure. In addition, it uses 6x and 14x fewer parameters than VDSR and DRCN, respectively, while obtaining better results. Contrary to DRCN, DRRN shares weight sets among residual units instead of one shared weight for all recursively-applied convolution layers. DRNN trains more stable and with deeper recursions than DRCN (52 in total) by emphasizing multi-path.

Inspired by DRCN, Tai et al. introduced MemNet (2017) [92]. The main contribution is a memory block consisting of recursive and gate units to mine persistent memory. The recursive unit is applied multiple times, similar to DRCN. The outputs are concatenated and sent to a gate unit, which is a simple 1x1 convolution layer. The adaptive gate unit controls the amount of prior information and the current state reserved. Fig. 9 visualizes MemNet. The effect of introducing gates was groundbreaking for sequence-to-sequence tasks (e.g., LSTM [106]) but deeply understanding its effect on SR tasks marks an open research question.

Inspired by DRRN, the authors of DSRN (2018) [97] explored a dual-state design with a multi-path network. It introduces two states, one operating in HR and one in LR space, which jointly exploits both LR and HR signals. The signals are exchanged recurrently between both spaces, LR-to-HR and HR-to-LR, via delayed feedback [107]. From LR-to-HR, it uses a transposed convolution layer to upsample. HR-to-LR is performed via strided convolutions. The final approximation uses the average of all estimations done in the HR space. Thus, it applies an extended formulation of iterative up-and-down upsampling (see Section 7.1). The two states use more parameters than DRRN, but less than DRCN. However, exploiting the dual-state design appropriately requires more exploration in the future.

The Super-Resolution Feedback Network (SRFBN, 2019) [64] is also using feedback [108]. The essential contribution is the feedback block (FB) as an actual recurrent cell. The FB uses multiple iterative up-and-downsamplings with dense skip connections to produce high-level discriminant features. The SRFBN generates a SR image for each iteration, and the FB block receives the previous iteration's output. It tries to generate the same SR image for single degradation tasks in each iteration. For more complex cases, it trains to return better and better quality images with each iteration via curriculum learning (see Section 6.1). SRFBN has shown significant improvement over the other frameworks but requires more research in the future.

Liu et al. proposed NLRN (2018) [109] that provides a non-local module to produce feature correlation for self-similarity. Each position in the image measures the feature correlation of each location in its neighborhood. NRLN utilizes adjacent recurrent stages between the feature correlation messages. And indeed, NLRN achieved slightly better performances than DRCN, DRCN, and MemNet.

Nevertheless, primary research on RNNs in SR is lately driven for MISR, such as video SR [110] or meta-learning [111] related tasks. In general, recurrent-based networks are interesting for saving parameters, but the major drawback is their computational overhead by repeatedly applying the same operation. Also, they are not parallelizable due to the time dependency. Alternatives are lightweight architectures, which are introduced next.

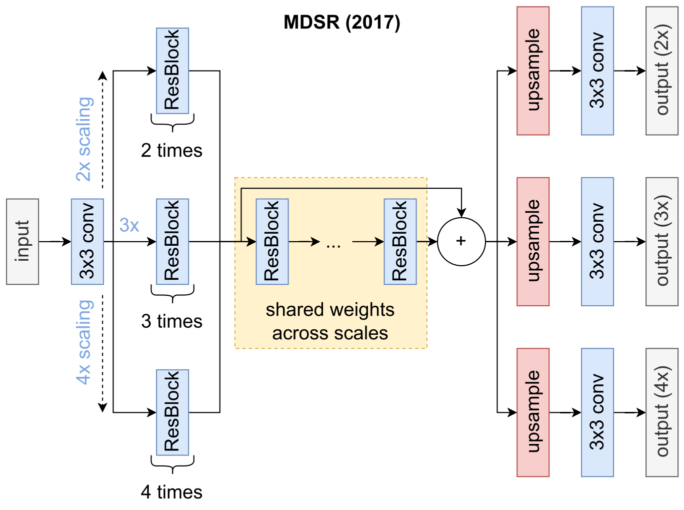

Fig. 10. Architecture design of MDSR [38]. It utilizes multi-path learning to select between multiple scaling factor paths (denoted by 2x, 3x, 4x on the bottom right of the blue boxes). Also, all paths share an intermediate feature extraction block to save parameters.

E. Lightweight Networks

So far, we have introduced models that increase the quality of SR images and a few that try to do the same, but with less computational effort. For instance, FSRCNN [33] was built to be faster than SRCNN [39] by utilizing smaller kernel sizes, post-upsampling, and 1x1 convolution layers for enhancing/reducing the channel dimension (see Section 7.2). Another example in this work was recurrent-based networks that reduce redundant parameters as described in Section 7.4. The downside of those lean recurrent networks is that parameter reduction comes at the expense of increased operations and inference time, an essential aspect of real-world scenarios. E.g., SR on mobile devices is limited by battery capacity, which depends on the computation power needed. Therefore, lightweight architectures explicitly focus on both execution speed and memory usage. Supplementary materials, available online, include a parameter comparison, and a fair comparison of execution speed is in welcome demand.

The MDSR (2017) [38] uses the multi-path approach to learn multiple scaling factors with shared parameters. It has three non-identical paths as pre-processing steps and three paths for upsampling. For a given scaling factor $s \in \{2, 3, 4\}$, MDSR is choosing deterministic between the three paths. The paths for larger scales are built deeper than those for lower scaling factors. Between the pre-processing and upsampling steps is a shared module consisting of multiple residual blocks. This feature extraction block is trained and commonly used for all scaling factors, visualized in Fig. 10. The main advantage is that one model is sufficient to train on multiple scales, which saves parameters and memory. In contrast, other SR models must be trained independently on different scales and saved independently for multi-scale applications. Nevertheless, adding a new scaling factor requires training from scratch.

Other lightweight architectures adapt this idea to enable parameter-efficient multi-scale training, such as CARN/CARN-M (2018) [112].

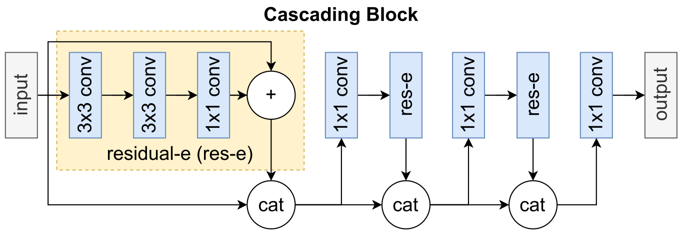

Fig. 11. Design of cascading blocks [112]. It is a densely connected block, which consists of Residual-E blocks [112] and 1x1 convolutions that take as input the Residual-E block's output and the residual connections from the layers before. The Residual-E block itself is built like a residual unit but performs group convolution, which is efficient in inference time, depending on the group size.

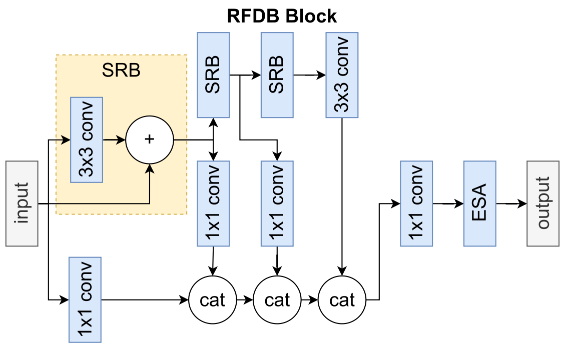

Fig. 12. Visualization of the RFDB block [114]. It uses a multi-path approach to distill features with 1x1 convolutions and introduces a shallow representation of the residual block, which consists of only one 3x3 convolution. In the end, all intermediate results are combined with one 1x1 convolution layer and an enhanced spatial attention [114] layer.

Moreover, it implements a cascading mechanism upon the residual network [103]. CARN consists of multiple cascading blocks (see Fig. 11) and 1x1 convolutions between them. The output of the cascading blocks is sent to all subsequent 1x1 convolutions like in the cascading block itself. Thus, the local cascading is almost identical to a global one. It allows multi-level representations and stable training like for residual networks. Ultimately, it chooses between three paths, which upsample the feature map to 2x, 3x, or 4x scaling factors via efficient sub-pixel layers similar to MDSR. Inspired by MobileNet [113], CARN also uses grouped convolution in each residual block component. This allows configuration of the model's efficiency since choosing different group sizes and the resulting performances are in a trade-off relationship. The residual blocks with group convolution can reduce the computation up to 14 times, depending on the group size. They tested a variant of CARN, which sets the group size so that the computation reduction is maximized and called it CARN-Mobile (CARN-M). Moreover, they further reduced CARN-M's parameters by enabling weight-sharing of their residual blocks within each cascading block (reduction by up to three times compared to non-shared).

Inspired by IDN and IMDB [115], the RFDN (2020) [114] rethinks the IMDB architecture by using RFDB blocks as shown in Fig. 12. RFDB blocks consist of feature distillation connections, which cascade 1x1 convolutions towards a final layer.

Moreover, it uses shallow residual blocks (SRBs), consisting of only one 3x3 convolution, to process a given input further. The final layer is a 1x1 convolution layer that combines all intermediate results. In the end, it applies the enhanced spatial attention [114], designed specifically for lightweight models. The RFDN architecture comprises subsequent RFDB blocks and uses a post-upsampling framework with a final sub-pixel layer.

A very hardware-aware and quantization-friendly network is XLSR (2021) [116]. It applies multi-paths to ease the burden of the convolution operations and employs 1x1 convolution to combine them pixel-wise. Each convolution layer has a small filter size (8, 16, 27). After the combination, it splits the feature map and applies multi-paths again. A core aspect of XLSR is the end activation layer, which exploits quantization benefits. Quantization is useful since it can save parameters by using more miniature bit representations [117]. Unfortunately, many mobile devices support 8-bit. Therefore, applying uint8 quantization on SR models that performed well in float32 or float16 does not work. Clipped ReLU (constrained to a max value of 1) as the last activation layer instead of the typical ReLU can dimish this issue. Nevertheless, the authors recommend searching for other maximum values with further experiments.

Generally, there are plenty of ideas to make SR models lightweight yet to be discovered. They include simplifications, quantization, and pruning of existing architectures. Also, more resource-constrained devices and applications utilizing SR are a growing field of interest [118].

F. Wavelet Transform-Based Networks

Different representations of images can bring benefits, such as computational speed up. The wavelet theory gives a stable mathematical foundation to represent and store multi-resolution images, depicting contextual and textural information [119]. Discrete Wavelet Transform (DWT) decomposes an image into a sequence of wavelet coefficients. The most frequent wavelet in SR is the Haar wavelet, computed via 2D Fast Wavelet Transform. The wavelet coefficients are calculated by repeating the decomposition to each output coefficient iteratively. It captures image details in four sub-bands: average (LL), vertical (HL), horizontal (LH), and diagonal (HH) information. One of the first networks that worked with wavelet prediction was DWSR (2017) [120]. It uses a simple network architecture to refine the differences between the LR and HR image wavelet decompositions in a pre-upsampling framework. First, it calculates the wavelet coefficients of the enlarged (with bicubic interpolation) LR image. Then it processes the wavelet coefficients with convolution layers. Next, it adds the initially calculated wavelet coefficients, which are beared with a residual connection. Thus, the convolution layers learn additional details of the coefficients. Finally, it applies the reverse process of 2D-DWT to obtain the SR image, as depicted in Fig. 13.

Another approach was proposed with WIDN (2019) [121], which uses stationary wavelet transform instead of DWT to perform better. A more sophisticated model was proposed around the same time as DWSR with Wavelet-SRNet (2017) [122]. It provides an embedding network, consisting of residual blocks, to generate feature maps from the LR image. Next, it applies DWT several times and utilizes multiple wavelet prediction networks. Finally, it applies the reverse process and uses a transposed convolution for upsampling. The coefficients are employed to a wavelet-based loss function, while the SR images are used for a traditional texture and MSE loss function. As a result, their network applies to different input resolutions with various magnifications and shows robustness toward unknown Gaussian blur, poses, and occlusions for MS-SR. The idea of multi-level wavelet CNNS can also be found in later publications, i.e., MWCNN (2018) [123].

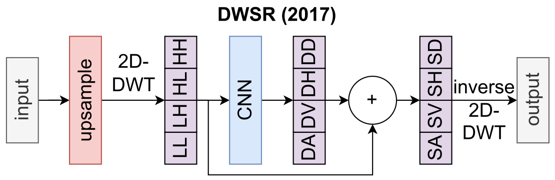

Fig. 13. Visualization of DWSR [120]. It calculates wavelet coefficients and uses a CNN to calculate the difference in the average (DA), vertical (DV), horizontal (DH), and diagonal (DD) sub-bands. Next, it adds the initially calculated coefficients via residual connection to the difference to derive the coefficients of the predicted SR image (SA, SV, SH, and SD), which is obtained by using the reverse 2D-DWT.

Following works apply a hybrid approach by mixing wavelet transform with other well-known SR methods. I.e., Zhang et al. proposed a wavelet-based SRGAN (2019) framework [124], which merged the advantages of SRGAN and wavelet decomposition. The generator uses an embedding network to process the input into feature maps, similar to Wavelet-SRNet. Next, it uses a wavelet prediction network to refine the coefficients, similar to DWSR. In 2020, Xue et al. [125] combined wavelets with residual attention blocks that contain channel attention and spatial attention modules (mixed attention, see the following Section 5) and called their network WRAN. Over the last several years, the application of wavelets also found its way to Video SR [126].

In general, wavelet transformations enable an efficient representation of images. As a result, SR models using this strategy often reduce the overall model size and computational costs while reaching similar performances to state-of-the-art architectures. However, this research area needs more exploration. For example, suitable normalization techniques since the distribution of high-frequency sub-bands and low-frequency sub-bands differ significantly or alternatives to convolution operations since they might be inappropriate due to the sparse representation of the high-frequency sub-bands.

VIII. Unsupervised Super-Resolution

The astonishing performance of supervised SR is imputed to their ability to learn natural image mainly from many LR-HR image pairs, mostly with known degradation mapping, which is often unknown in practice. Thus, supervised trained SR models are sometimes unreliable for practicable use-cases. For instance, when the training dataset has LR images generated without anti-aliasing (high-frequency is preserved), then the SR model trained on that dataset is not adequately suitable for real-world LR images generated with anti-aliasing (smoothed images). In addition, some specialized application areas lack LR-HR image pair datasets. Therefore, there is a growing interest in unsupervised SR. We examine briefly this field, for further reading inclusive flow-based methods (density estimation of degradation kernels) we refer to the survey of Liu et al. [127].

A. *Weakly-Supervised*

Weakly-supervised methods use unpaired LR and HR images like WESPE (2018) [128]. WESPE consists of two generators and two discriminators. The first generator takes a LR image and super-resolves it. The output of the first generator constitutes a SR image, but is also regularized with TV loss [59]. The second generator takes the prediction of the first generator and performs the inverse mapping. The result of the second generator is optimized via content loss [12] with the original input, the LR image. The two discriminators take the SR image of the first generator and are trained to distinguish between predictions and original HR images. The first discriminator classifies inputs based on image colors into SR or HR images. The second discriminator uses image textures [61] in order to classify.

A similar approach is a cycle-in-cycle SR framework called CinCGAN (2018) [52], based on CycleGAN [129]. It uses a total of four generators and two discriminators. The first generator takes a noisy LR image and maps it to the clean version. The first discriminator is trained to distinguish between clean LR images from the dataset and the predicted clean images. The second generator trains the inverse function. Thus, it generates the noisy image from the predicted clean version, which closes the first cycle of a CycleGan. The third generator is of particular interest because it is the actual SR model which upsamples the LR image to HR. The second discriminator is trained to distinguish between the predicted and the dataset's HR images. The last generator maps the predicted HR image to the noisy LR image, which closes the second cycle of a CycleGAN. Besides its promising results and similar approaches [130], it requires further research to decrease learning difficulty and computational cost.

B. *Zero-Shot*

Zero-shot or one-shot learning is associated with training on objects and testing on entirely different objects from a different class that was never observed. Ideally, a classifier trained on horses should recognize zebras if the knowledge "zebras look like striped horses" is transferred [132]. The first publication on Zero-Shot in SR is ZSSR (2017) [87]. The goal was to train only on one image at hand, one of a kind. The degradation mapping for ZSSR was chosen to be fixed, such as bicubic. ZSSR downsamples the LR image and trains a CNN to reverse the degradation mapping. The trained CNN is then finally used directly on the LR image. Surprisingly, this method reached better results than SRCNN and was close to VDSR.

Upon this, a Degradation Simulation Network (DSN, 2020) [133] based on depth information [134] was proposed to avoid a pre-defined degradation kernel. It uses bi-cycle training to simultaneously learn the unknown degradation kernel and the reconstruction of SR images. The MZSR [135] merged the ZSSR setting with meta-learning and used an external dataset to learn different blur kernels, which is called task distribution in the field of meta-learning. The SR model is then trained on the downsampled image similar to ZSSR with the blur kernel returned from the meta-test phase. The profit of this approach is that it conducts the SR model to learn specific information faster and performs better than pure ZSSR. Zero-shot learning for SR marks an exciting area for further research because it is highly practical, especially for applications where application-specific datasets are rare or non-existent.

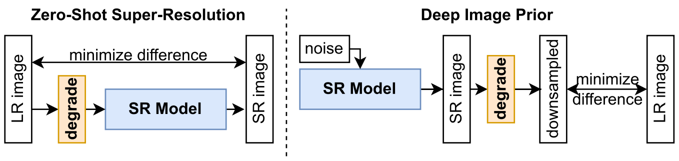

Fig. 14. Zero-Shot Super-Resolution (ZSSR) [87] and Deep Image Prior (DIP) [131]. ZSSR uses the LR image and downsamples it and the SR model learns to reverse the downsampling. For the final prediction of the SR image of the LR image, it is applied to the LR image directly. DIP uses fixed noise as input and predicts the SR image, which is downsampled to optimize the difference between the downsampled image and a given LR image. The final prediction uses the SR model to predict the SR image but skips the degradation mapping.

C. *Deep Image Prior*

Ulyanov et al. [131] proposed Deep Image Prior (DIP), which contradicts the conventional paradigm of training a CNN on large datasets. It uses a CNN to predict the LR image when downsampled, given some random noise instead of an actual image. Therefore, it follows the strategy of ZSSR by using only the LR image. However, it fixes the input to random noise and applies a fixed downsampling method to the prediction. Moreover, it optimizes the difference between the downsampled prediction and the LR image. The CNN then produces the SR image without using the fixed downsampling method. Thus, it generates an SR image out of noise instead of transforming a raw image. The difference between ZSSR and DIP is highlighted in Fig. 14. Surprisingly, the results were close to LapSRN [136]. Unfortunately, it is a theoretical publication about image priors, and the approach is too slow to be useful for most practical applications, as the authors stated themselves. However, it does not exclude future ideas that could enhance the practicability of DIP concerning better image reconstruction quality and especially runtime.

IX. Neural Architecture Search

Recently, a new field called Neural Architecture Search (NAS) has gained popularity, which aims to automatically derive designs instead of hand-designed Neural Networks (NNs) created by human artistry [10]. Fortunately, NAS can be applied to derive parts of SR models.

MoreMNAS (2019) [10] investigated NAS for SR and resource-aware mobile devices. The goal is to derive cell-based designs similar to the idea of residual blocks. A SR model

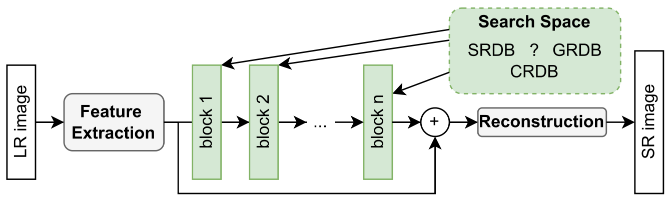

Fig. 15. Visualization of ESRN [137]. A controller derives an architecture by selecting from a pre-set of various efficient residual blocks.

usually consists of feature extraction, non-linear mapping, and reconstruction, like in SRCNN. Their search space is constrained to non-linear mapping. They modified the NAS algorithm NS-GANet [138] to SR, which is based on the evolution algorithm NSGA-II with multi-objective evolution [139]. In the case of MoreMNAS, multi-objective fitness is defined by high PSNR, low FLOPS, and a few parameters. Furthermore, they combine it with reinforcement learning, in which an LSTM [106] controller returns design choices based on NSGA-II, and the evaluator returns a MSE-based reward. The evaluator trains and evaluates the design choices in a separate training phase. As a result, MoreMNAS delivers rivaling models compared to methods (like FSRCNN, VDSR, DRRN, CARN-M, and DRCN) with fewer FLOPS. FALSR (2019) [140] is a similar approach, which differs by using a hybrid controller and a cell-based elastic search space that enables both macro (the connections among different cell blocks) and micro search (cell blocks). A faster method with comparable performance to FALSR was proposed with ESRN (2019) [137]. They search locations of pooling and upsampling operators and derive the architecture with the guidance of block credits, which weighs the sampling probability of mutation to favor admirable blocks. Moreover, they constrain the search space to efficient residual dense blocks from known lightweight architectures, such as RFDB from RFDN (see Section 7.5). Fig. 15 shows the latter. DeCoNAS (2021) [141] is a similar approach, which uses ENAS [142] and densely connected network blocks.

HNAS (2020) [143] introduces a hierarchical search space that consists of a cell-level search space and a network-level search space, similar to FALSR. The cell-level search space identifies a series of blocks that increase model capacity. The network-level search space determines the position of upsampling layers. For cell-level search space, HNAS uses two LSTM controllers: one derives normal cells consisting of convolutions, and the other one the upsampling cells (bilinear interpolation, sub-pixel layer, transposed convolution, and more). On top, a network-level controller is used to select the positions of the upsampling cells.

NAS-DIP (2020) [144] combines NAS with DIP [131] (see Section 8.3 and supplementary material, available online, for visualization). It consists of two phases: First, they apply reinforcement learning with a RNN controller (PSNR as a reward) to generate a network structure. The second phase uses the network structure and optimizes the mapping from random noise to a degenerated image (as in DIP [131]). Next, it repeats the phases with the reward gained after the second phase. Their search space is designed for two components. The first component is the upsampling cell (e.g., bilinear, bicubic, nearest-neighbor). The second component is feature extraction (e.g., convolution, depth-wise convolution, and others). Moreover, they extend their search space to learn cross-level connection patterns across various feature levels in an autoencoder network close to the U-Net of DIP.

Another way of combining NAS with frameworks that worked well in SR was proposed by Lee et al. (2020) [145]. They combined NAS with GANs and defined search spaces for generators and discriminators. The GAN framework is then used to train the generator in the manner of SRGAN. However, they faced substantial stability problems during their work, which needs further research.

One major drawback of reinforcement learning based NAS is that they need additional training to evaluate design choices. It imposes a significant time overhead on the search process. Another problem is that reinforcement learning based NAS approaches rely on discrete choices performed and exploited by a controller. One way to omit that is by using a gradient-based search like DARTS [146]. HiNAS (2021) [147] adopts gradient-based search and builds a flexible hierarchical search space. The search space is similar to HNAS, which refers to micro and macro search. During cell search (micro), HiNAS considers all combinations of operations as a weighted sum. The highest weights derive the final cell design. The macro search follows the same idea as micro search and conducts several supercells with different settings and derives the final architecture gradient-based. Wu et al. (2021) [148] proposed something similar. They adopt gradient-based search but extend the search space into three levels. The first level describes the network level, which defines all candidate network paths. The second level determines possible candidate operations. Lastly, the kernel-level is a subset of convolution kernel dimensions.

The main streams of NAS in SR are dominated by evolutionary algorithms paired with reinforcement learning. They imply a significant search time, which is inevitable due to the repetitive action and reward processes. Most recently, gradient-based methods were applied to facilitate the search time. Moreover, they enable continuous search by relaxing the search space, transforming discrete choices into a weighted sum of possible paths. The trend points to various hierarchies within the search process, meaning that more aspects of determining a network design for SR are incorporated into NAS. A well-designed architecture is critical for success on SR tasks. NAS approaches have yet to outperform hand-crafted state-of-the-art architectures. More elaborate methods must be introduced to produce better-performing architectures concerning quality and inference speed.

X. Discussion and Future Directions

This section gives a short overview of the topics discussed in this paper and shows potential future directions.

IQA Evaluating quality of generated images is difficult and still an open problem. PSNR and SSIM are valuable metrics because of fast calculation, but do not always match subjectively perceived quality. Developing a metric that overcomes this problem is of high interest. Future research should focus on such a metric since it would also be interesting besides SR, e.g., Text-to-Image Synthesis [51].

Learning Objectives. The selection of learning objectives strongly depends on the data domain. A learning objective that fits all SR domains is not given and remains open for research. Most SR papers use simple formulations, e.g., regression-based or combinations of them. Exploring various probable loss functions is highly demanded. Recent developments in uncertainty-driven loss functions can be a promising direction for new loss functions.

Upsampling. One problem with the most commonly used upsampling layers is that they produce artifacts. Also, the scale factor must be predefined. Therefore, using them in an application like zooming is not feasible. An alternative is meta-upscaling, which enables arbitrary scaling but comes with computational overhead and stability issues. Thus, new lightweight layers for arbitrary upscaling are highly interesting and should be focused on in the following years. Moreover, bilateral upsampling and guided upsampling play a crucial role in nowadays applications, which should be explored more in the future concerning deep learning.

Unsupervised Super-Resolution. Data-rich datasets inherently dictate the overall performance of any SR method. However, LR-HR image pairs are not always given. In such a case, unsupervised SR methods are interesting. It marks an exciting area for further research because it is highly practical, especially for applications where datasets are rare or non-existent. Zero-shot learning and DIP are good starting points, but the execution speed and practicability is not enough for real-world applications. New ideas are required to enhance practicability regarding better image reconstruction quality and especially execution time.

Neural Network Architectures. Concerning the statement of DIP, a well-designed architecture is critical for success on SR tasks. Recently, many architectures were proposed to investigate certain aspects, such as SRResNet [124] (handling vanishing/exploding gradients), DSRN [97] (exploring recurrence for SR), XLSR [116] (hardware-aware architecture) and SRGAN [124] (exploring GANS for SR). Future work has to investigate which modules (like residual blocks or attention mechanisms) contribute the most regarding certain aspects to formulate a guideline for engineers on constructing a suitable architecture for a given problem: high quality versus computational efficiency.

XI. Conclusion

With the advent of DL, Super-Resolution (SR) has recently become a rapidly moving research area. However, despite promising results, the field continues to face challenges that call for more research, e.g., flexible upsampling. We reviewed the area of SR with recent advances and examined state-of-the-art models, such as transformer-based SR, and other architecture designs proposed lately (e.g., denoising diffusion probabilistic models). We critically examined current strategies and identified new research areas. We complemented previous surveys by incorporating the latest developments and ideas, such as uncertainty-driven loss functions, new normalization techniques, wavelet networks, or neural architecture search. Also, we added various visualizations of models and methods in the chapters and the supplementary material, available online, to make navigation through this domain more accessible. We hope this review helps researchers to push the boundaries of DL applied to SR further.

References

[1] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surf.*, Springer, 2010, pp. 711–730.

[2] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE 8th Int. Conf. Comput. Vis.*, 2001, pp. 416–423.

[3] D. Valsesia and E. Magli, "Permutation invariance and uncertainty in multitemporal image super-resolution," 2021, *arXiv:2105.12409*.

[4] S. M. A. Bashir, Y. Wang, M. Khan, and Y. Niu, "A comprehensive review of deep learning-based single image super-resolution," *PeerJ Comput. Sci.*, vol. 7, 2021, Art. no. e621.

[5] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3365–3387, Oct. 2021.

[6] S. Anwar and N. Barnes, "Densely residual Laplacian super-resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1192–1204, Mar. 2022.

[7] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Trans. Image Process.*, vol. 29, pp. 4027–4040, 2020.

[8] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 372–386.

[9] D. Thapa, K. Raahemifar, W. R. Bobier, and V. Lakshminarayanan, "A performance comparison among different super-resolution techniques," *Comput. Elect. Eng.*, vol. 54, pp. 313–329, 2016.

[10] X. Chu, B. Zhang, and R. Xu, "Multi-objective reinforced evolution in mobile neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 99–113.

[11] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Trans. Comput. Imag.*, vol. 2, no. 2, pp. 109–122, Jun. 2016.

[12] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "EnhanceNet: Single image super-resolution through automated texture synthesis," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4491–4500.

[13] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4681–4690.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[15] N. Ponomarenko et al., "Image database TID2013: Peculiarities, results and perspectives," *Signal Process.: Image Commun.*, vol. 30, pp. 57–77, 2015.

[16] J. Kim and S. Lee, "Deep learning of human visual sensitivity in image quality assessment framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1676–1684.

[17] H. Talebi and P. Milanfar, "NIMA: Neural image assessment," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3998–4011, Aug. 2018.

[18] K. Ma, W. Liu, T. Liu, Z. Wang, and D. Tao, "dipIQ: Blind image quality assessment by learning-to-rank discriminable image pairs," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3951–3964, Aug. 2017.

[19] C. Burges et al., "Learning to rank using gradient descent," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 89–96.

[20] T.-Y. Liu, "Learning to rank for information retrieval," 2011.

[21] X. Liu, J. Van De Weijer, and A. D. Bagdanov, "RankIQA: Learning from rankings for no-reference image quality assessment," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1040–1049.

[22] G. Koch et al., "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, Lille, 2015.

[23] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[25] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2567–2581, May 2022.

[26] W. Xue, L. Zhang, X. Mou, and A. C. Bovik, "Gradient magnitude similarity deviation: A highly efficient perceptual image quality index," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 684–695, Feb. 2014.

[27] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[28] R. Reisenhofer, S. Bosse, G. Kutyniok, and T. Wiegand, "A haar wavelet-based perceptual similarity index for image quality assessment," *Signal Process.: Image Commun.*, vol. 61, pp. 33–43, 2018.

[29] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. IEEE 37th Asilomar Conf. Signals Syst. Comput.*, 2003, pp. 1398–1402.

[30] B. Zhang, P. V. Sander, and A. Bermak, "Gradient magnitude similarity deviation on multiple scales for color image quality assessment," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 1253–1257.

[31] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.

[32] Y. Matsui et al., "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 21 811–21 838, 2017.

[33] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 391–407.

[34] *MATLAB version 9.3.0.713579 (R2017b)*, The Mathworks, Inc., Natick, MA, USA, 2017.

[35] A. Lugmayr, M. Danelljan, and R. Timofte, "NTIRE 2020 challenge on real-world image super-resolution: Methods and results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 494–495.

[36] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, "The 2018 PIRM challenge on perceptual image super-resolution," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 334–355.

[37] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 286–301.

[38] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 136–144.

[39] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[40] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4799–4807.

[41] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-SR: A magnification-arbitrary network for super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1575–1584.

[42] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.

[43] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 723–731.

[44] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 624–632.

[45] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?," 2017, *arXiv: 1703.04977*.

[46] Q. Ning, W. Dong, X. Li, J. Wu, and G. Shi, "Uncertainty-driven loss for single image super-resolution," *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 16398–16409.

[47] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[48] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140.

[49] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2305–2318, Oct. 2019.

[50] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[51] S. Frolov, T. Hinz, F. Raue, J. Hees, and A. Dengel, "Adversarial text-to-image synthesis: A review," 2021, *arXiv:2101.09983*.

[52] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 701–710.

[53] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, "A fully progressive approach to single-image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 864–873.

[54] L. Yu, X. Long, and C. Tong, "Single image super-resolution based on improved WGAN," in *Proc. Int. Conf. Adv. Control Automat. Artif. Intell.*, Shenzhen, China, 2018, pp. 21–22.

[55] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," 2017, *arXiv:1704.00028*.

[56] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? A large-scale study," 2017, *arXiv:1711.10337*.

[57] K. Singla, R. Pandey, and U. Ghanekar, "A review on single image super resolution techniques using generative adversarial network," *Optik*, vol. 266, 2022, Art. no. 169607.

[58] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[59] M. Vella and J. F. Mota, "Single image super-resolution via CNN architectures and TV-TV minimization," 2019, *arXiv:1907.05380*.

[60] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 49–70, 2000.

[61] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 262–270.

[62] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020, *arXiv:2006.11239*.

[63] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," 2021, *arXiv:2104.07636*.

[64] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, "Feedback network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3867–3876.

[65] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3147–3155.

[66] W. Shi et al., "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1874–1883.

[67] W. Shi et al., "Is the deconvolution layer the same as a convolutional layer?," 2016, *arXiv:1609.07009*.

[68] Z. Wojna et al., "The devil is in the decoder: Classification, regression and GANs," *Int. J. Comput. Vis.*, vol. 127, no. 11, pp. 1694–1706, 2019.

[69] S. Kundu, H. Mostafa, S. N. Sridhar, and S. Sundaresan, "Attention-based image upsampling," 2020, *arXiv:2012.09904*.

[70] Y. Jo and S. J. Kim, "Practical single-image super-resolution using look-up table," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 691–700.

[71] C. Peng, W.-A. Lin, H. Liao, R. Chellappa, and S. K. Zhou, "SAINT: Spatially aware interpolation network for medical slice synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7750–7759.

[72] L. Chai, M. Gharbi, E. Shechtman, P. Isola, and R. Zhang, "Any-resolution training for high-resolution image synthesis," 2022, *arXiv:2204.07156*.

[73] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[74] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[75] Y. Yang and Y. Qi, "Image super-resolution via channel attention and spatial graph convolutional network," *Pattern Recognit.*, vol. 112, 2021, Art. no. 107798.

[76] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.

[77] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," 2019, *arXiv:1903.10082*.
[78] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using swin transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 1833–1844.
[79] D. Zhang, F. Huang, S. Liu, X. Wang, and Z. Jin, "SwinFIR: Revisiting the SwinIR with fast Fourier convolution and improved training for image super-resolution," 2022, *arXiv:2208.11247*.
[80] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11 065–11 074.
[81] H. Zhao, X. Kong, J. He, Y. Qiao, and C. Dong, "Efficient image super-resolution using pixel attention," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 56–72.
[82] B. Niu et al., "Single image super-resolution via a holistic attention network," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 191–207.
[83] F. Wang, H. Hu, and C. Shen, "BAM: A lightweight and efficient balanced attention mechanism for single image super resolution," 2021, *arXiv:2104.07566*.
[84] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.
[85] Y. Bei, A. Damian, S. Hu, S. Menon, N. Ravi, and C. Rudin, "New techniques for preserving global structure and denoising with low information loss in single-image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 874–881.
[86] N. Ahn, B. Kang, and K.-A. Sohn, "Image super-resolution via progressive cascading residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 791–799.
[87] A. Shocher, N. Cohen, and M. Irani, ""Zero-shot" super-resolution using deep internal learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3118–3126.
[88] H. Ren, M. El-Khamy, and J. Lee, "Image super resolution based on fusing multiple convolution neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 54–61.
[89] K. Urazoe, N. Kuroki, Y. Kato, S. Ohtani, T. Hirose, and M. Numa, "Multi-category image super-resolution with convolutional neural network and multi-task learning," *IEICE Trans. Inf. Syst.*, vol. 104, no. 1, pp. 183–193, 2021.
[90] Y. Shi, K. Wang, C. Chen, L. Xu, and L. Lin, "Structure-preserving image super-resolution via contextualized multitask learning," *IEEE Trans. Multimedia*, vol. 19, no. 12, pp. 2804–2815, Dec. 2017.
[91] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 448–456.
[92] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4539–4547.
[93] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3883–3891.
[94] J. Liu, J. Tang, and G. Wu, "Adadm: Enabling normalization for image super-resolution," 2021, *arXiv:2111.13905*.
[95] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Models Image Process.*, vol. 53, no. 3, pp. 231–239, 1991.
[96] C. Tan, L. Wang, and S. Cheng, "Image super-resolution via dual-level recurrent residual networks," *Sensors*, vol. 22, no. 8, 2022, Art. no. 3058.
[97] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, and T. S. Huang, "Image super-resolution via dual-state recurrent networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1654–1663.
[98] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.
[99] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5353–5360.
[100] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 519–534.
[101] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2802–2810.
[102] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, Springer, 2015, pp. 234–241.
[103] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[104] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[105] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1637–1645.
[106] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[107] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 2067–2075.
[108] A. R. Zamir et al., "Feedback networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1308–1317.
[109] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," 2018, *arXiv:1806.02919*.
[110] T. Isobe, X. Jia, S. Gu, S. Li, S. Wang, and Q. Tian, "Video super-resolution with recurrent structure-detail network," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 645–660.
[111] S. Park, J. Yoo, D. Cho, J. Kim, and T. H. Kim, "Fast adaptation to super-resolution networks via meta-learning," in *Proc. 16th Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 754–769.
[112] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 252–268.
[113] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[114] J. Liu, J. Tang, and G. Wu, "Residual feature distillation network for lightweight image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 41–55.
[115] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 2024–2032.
[116] M. Ayazoglu, "Extremely lightweight quantization robust real-time single-image super resolution for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2472–2479.
[117] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.
[118] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Comput. Sci.*, vol. 2, no. 6, pp. 1–20, 2021.
[119] M. Stephane, "A wavelet tour of signal processing," 1999.
[120] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep wavelet prediction for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 104–113.
[121] F. Sahito, P. Zhiwen, J. Ahmed, and R. A. Memon, "Wavelet-integrated deep networks for single image super-resolution," *Electronics*, vol. 8, no. 5, 2019, Art. no. 553.
[122] H. Huang, R. He, Z. Sun, and T. Tan, "Wavelet-SRNet: A wavelet-based CNN for multi-scale face super resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1689–1697.
[123] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-CNN for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 773–782.
[124] Q. Zhang et al., "Super-resolution reconstruction algorithms based on fusion of deep learning mechanism and wavelet," in *Proc. 2nd Int. Conf. Artif. Intell. Pattern Recognit.*, 2019, pp. 102–107.
[125] S. Xue, W. Qiu, F. Liu, and X. Jin, "Wavelet-based residual attention network for image super-resolution," *Neurocomputing*, vol. 382, pp. 116–126, 2020.
[126] X. Zhu, Z. Li, J. Lou, and Q. Shen, "Video super-resolution based on a spatio-temporal matching network," *Pattern Recognit.*, vol. 110, 2021, Art. no. 107619.
[127] A. Liu, Y. Liu, J. Gu, Y. Qiao, and C. Dong, "Blind image super-resolution: A survey and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Aug. 30, 2022, doi: 10.1109/TPAMI.2022.3203009.
[128] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool, "WESPE: Weakly supervised photo enhancer for digital cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 691–700.

[129] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.

[130] A. Bulat, J. Yang, and G. Tzimiropoulos, "To learn image super-resolution, use a GAN to learn how to do image degradation first," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 185–200.

[131] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9446–9454.

[132] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2251–2265, Sep. 2019.

[133] X. Cheng, Z. Fu, and J. Yang, "Zero-shot image super-resolution with depth guided internal degradation learning," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 265–280.

[134] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3828–3838.

[135] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3516–3525.

[136] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep Laplacian pyramid networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2599–2613, Nov. 2019.

[137] D. Song, C. Xu, X. Jia, Y. Chen, C. Xu, and Y. Wang, "Efficient residual dense block search for image super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 12 007–12 014.

[138] Z. Lu et al., "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Computation Conf.*, 2019, pp. 419–427.

[139] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proc. 13th Int. Conf. Intell. Syst. Appl. Power Syst.*, 2005, pp. 84–91.

[140] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," in *Proc. IEEE 25th Int. Conf. Pattern Recognit.*, 2021, pp. 59–64.

[141] J. Y. Ahn and N. I. Cho, "Neural architecture search for image super-resolution using densely constructed search space: DeCoNAS," in *Proc. IEEE 25th Int. Conf. Pattern Recognit.*, 2021, pp. 4829–4836.

[142] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 4095–4104.

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

[144] Y.-C. Chen, C. Gao, E. Robb, and J.-B. Huang, "NAS-DIP: Learning deep image prior with neural architecture search," in *Proc. 16th Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 442–459.

[145] R. Lee et al., "Journey towards tiny perceptual super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 85–102.

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

[147] H. Zhang, Y. Li, C. Gong, H. Chen, Z. Bai, and C. Shen, "Memory-efficient hierarchical neural architecture search for image restoration," 2020, *arXiv:2012.13212*.

[148] Y. Wu, Z. Huang, S. Kumar, R. S. Sukthanker, R. Timofte, and L. Van Gool, "Trilevel neural architecture search for efficient single image super-resolution," 2021, *arXiv:2101.06658*.

[149] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10 012–10 022.

Brian B. Moser received the MSc degree in computer science from the TU Kaiserslautern, in 2021. He is currently working toward the PhD degree with the TU Kaiserslautern and research assistant with the German Research Center for Artificial Intelligence (DFKI) in Kaiserslautern. His research interests include image super-resolution and deep learning.

Federico Raue received the MSc degree in artificial intelligence from Katholieke Universiteit Leuven, in 2005, and the PhD degree from TU Kaiserslautern, in 2018. He is a senior researcher with the German Research Center for Artificial Intelligence (DFKI) in Kaiserslautern. His research interests include meta-learning and multimodal machine learning.

Stanislav Frolov received the MSc degree in electrical engineering from the Karlsruhe Institute of Technology, in 2017. He is currently working toward the PhD degree with the TU Kaiserslautern and research assistant with the German Research Center for Artificial Intelligence (DFKI) in Kaiserslautern. His research interests include generative models and deep learning.

Sebastian Palacio is a researcher in machine learning and head of the Multimedia Analysis and Data Mining Group, German Research Center for Artificial Intelligence (DFKI). His PhD topic was about explainable AI with applications in computer vision. Other research interests include adversarial attacks, multi-task, curriculum, and self-supervised learning.

Jörn Hees received the PhD degree from the TU Kaiserslautern, in 2018 on the topic of simulating human associations with linked data. He is a professor for data science with the University of Applied Sciences Bonn-Rhein-Sieg. His research interests include machine learning, knowledge graphs, and multimedia analysis.

Andreas Dengel is a professor with the Department of Computer Science, TU Kaiserslautern and executive director of the German Research Center for Artificial Intelligence (DFKI) in Kaiserslautern, head of the Smart Data and Knowledge Services Research Area at DFKI and of the DFKI Deep Learning Competence Center. His research focuses on machine learning, pattern recognition, quantified learning, data mining, semantic technologies, and document analysis.