Ultra Resolution Generative Adversarial Network for Perceptually Realistic Image Super-Resolution

Moein Esfahani
esfahani.moein@gmail.com

Abstract

This paper presents an advanced image superresolution system that utilizes a two-stage training approach to generate high-resolution images with enhanced perceptual quality. The system employs a residual-in-residual-density block network (RRDBNet) as the generator, initially trained to minimize the pixel-wise reconstruction error. Subsequently, the generator is fine-tuned using a Generative Adversarial Network (GAN) framework, incorporating perceptual and adversarial losses to improve texture detail and overall visual fidelity. This methodology addresses the inherent challenge of balancing quantitative accuracy with perceptual realism in super-resolution. The details of the implementation, including network architecture, loss functions, training strategies, and optimization techniques, are discussed. The system demonstrates promising results, achieving a balance between high PSNR/SSIM metrics and visually compelling image generation. code is available: Github

1. Introduction

the task of reconstructing high-resolution (HR) images from their low-resolution (LR) counterparts, has become increasingly important in various applications, from medical imaging to surveillance. Deep learning techniques, particularly Generative Adversarial Networks (GANs), have revolutionized this field, offering the potential to generate perceptually pleasing HR images with intricate details [2]. However, a fundamental challenge persists: achieving a satisfactory balance between pixel-level accuracy, which ensures fidelity to the original image structure, and perceptual quality, which focuses on the visual realism and plausibility of the generated details.

Existing state-of-the-art SR methods, including those based on GANs like SRGAN [2] or its next generation as ESRGAN [5], often face a trade-off. Although they excel at generating visually appealing textures and fine details, they can sometimes introduce distortions or artifacts that deviate from the true underlying structure of the original HR image. In contrast, models optimized solely for pixel-wise reconstruction tend to produce smoother, less detailed results that lack visual richness. The core problem, therefore, lies in developing a robust SR framework that can simultaneously preserve structural fidelity and generate perceptually convincing high-frequency details without introducing unwanted artifacts.

2. Related Works

The field of image super-resolution has a history, evolving from traditional interpolation methods to deep learning architectures. Early approaches relied on techniques like bicubic interpolation, which produce blurry and detail-lacking results. More advanced methods incorporated statistical models and neighbor embedding techniques to infer high-frequency details. However, these methods often struggled with complex textures and large upscaling factors. Convolutional Neural Networks (CNNs) demonstrated remarkable capabilities in learning complex mappings from LR to HR images. Pioneering work like SRCNN [1] showed results in SR.

Then, the GANs brought about a paradigm shift by focusing on perceptual realism. SRGAN [2] utilized an adversarial loss to train a generator network to produce photorealistic HR images. ESRGAN further optimized the model by introducing the Residual-in-Residual Dense Block (RRDB) architecture and improved adversarial and perceptual losses. These innovations led to significant improvements in both perceptual quality and the generation of realistic textures. Numerous subsequent works have built upon the foundation of ESRGAN, exploring different network architectures, loss functions, and training strategies to further enhance SR performance. These include investigations into multiscale processing, attention mechanisms, and the incorporation of prior knowledge. Currently, state-of-the-art models are mainly difusion models and one of the recent ones are the resShift model [6].

Building upon the advancements of real-ESRGAN [4], this paper introduces an advanced SR system designed to explicitly address this challenge. Our approach leverages a novel two-stage training strategy similar to ESRGAN. We tried to modify the loss function that better balances texture

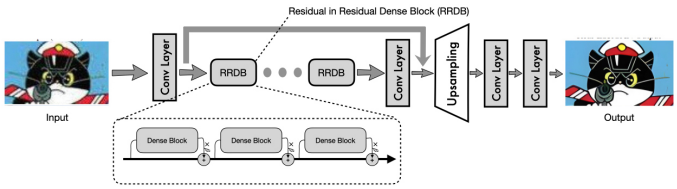

Figure 1. Generator Architecture of the Model, combination of residual layers and Dense Blocks

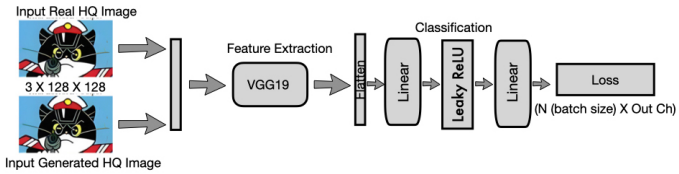

Figure 2. Discriminator architecture of the model, using pretrained VGG19 as feature extraction method.

generation and structural preservation with new data augmentation strategies in the latest version of Pytorch tailored for preserving fine details.

This paper will delve into the specific architectural and hyperparameter choices. Furthermore, we will critically analyze the trade-offs inherent in different evaluation metrics and discuss potential strategies for deploying such a system in practical, real-world scenarios.

3. Materials and Methodology

3.1. Datasets

The training dataset consisted of HQ images processed to create matched low-resolution (LR) and high-resolution (HR) pairs. Training Images: 2,385 diverse high-resolution images. Dataset is publicly available as Div2X and Flicker 2K dataset [3, 7].then, for training the model we applied Patch Extraction where 128×128 HR patches with corresponding 32×32 LR patches were extracted. for generalizability and data Augmentation we used latest version of PyTorch for Random cropping and horizontal flipping were employed for data augmentation. Important for latest versions which optimized for dataset class and data loader. LR images were generated from HR images using bicubic downsampling with antialiasing. in contrast to real-ESRGAN [4] we applied LR processing seperately than training process, as part of training optimization. This methodology of using random cropping and augmentation from diverse image sizes has several benefits for model generalization, including content diversity, scale invariance, content-adaptive learning, and detail preservation.

3.2. Model Architecture

The implemented super resolution system utilizes a two-stage training approach **Stage 1:** RRDBNet Training - Focuses on minimizing pixel-wise reconstruction error. and **Stage 2:** USRGAN Training - Improves perceptual quality through adversarial and perceptual losses.

Generator Network (RRDBNet) The generator network in 1 adopts a Residual-in-Residual Dense Block (RRDB) architecture. This multi-part architecture commences with an Input Layer accepting Low-Resolution (LR) images at a size of 32×32 pixels. Subsequently, Feature Extraction is performed by an Initial convolutional layer employing 64 filters. The core of the generator lies in its RRDB Blocks, where 23 such blocks with dense connections are stacked. These blocks strategically integrate dense connections internally, implement residual learning across various levels, and then applies batch normalization to promote stable training. Following the RRDB blocks, Upsampling is achieved through two pixel-shuffle layers, facilitating a 4× increase in resolution. Finally, the Output Layer generates High-Resolution (HR) images with dimensions of 128×128 pixels.

Discriminator Network The discriminator network in 2 has VGG-style architecture. This architecture's Input stage receives both 128×128 High-Resolution (HR) ground truth images and the generated Super-Resolved (SR) images. Next, Feature Extraction is carried out through a Series of convolutional layers, systematically increasing the number of channels (64→128→256→512). Following feature extraction, Classification is performed by Dense layers, ultimately leading to a binary prediction of real/fake. Throughout the network, Leaky ReLU activation functions are employed, consistently using a slope of $\alpha = 0.2$. Furthermore, Spectral Normalization is Applied across the network to enhance the stability of the GAN training process.

4. Experiments

The training process for the SR system was conducted in two distinct stages. This two-stage approach aimed to optimize both reconstruction accuracy and perceptual quality.

Stage 1: RRDBNet Training: In the first stage, the RRDBNet generator, as depicted in Figure 1, was trained with the goal of minimizing pixel-wise reconstruction error. For this stage, Pixel Loss (L_1) served as the loss function. This loss function measured the absolute pixel-wise differences between the generated SR images and the corresponding HR ground truth images, thereby encouraging structural fidelity and aiming to achieve a high PSNR. The Adam optimizer was used to update the generator's weights. The initial learning rate for the Adam optimizer was set to $1e-4$. The RRDBNet generator did the training for 200 epochs. The final output of this stage producing structurally

accurate upscaled images, although these images exhibited some degree of blurriness.

Stage 2: USRGAN Training: In the second stage, the generator, pre-trained in Stage 1, was further fine-tuned using a GAN-based approach. This fine-tuning aimed to enhance the perceptual quality of the generated SR images. For Initialization, the generator network's weights were initialized using the weights obtained from the pre-trained RRDBNet in Stage 1. In this stage, a combined loss function was employed to balance pixel-wise accuracy, perceptual quality, and adversarial realism. This combined loss consisted of Pixel Loss (L_1), Content Loss, and Adversarial Loss, each detailed as follows: Pixel Loss (L_1): While still in use, the weight of the pixel loss was reduced to allow the network to prioritize perceptual quality to a greater extent. Content Loss (VGG): To assess perceptual differences between the generated SR images and the HR images, feature maps from the VGG19 network were computed. This loss encouraged both semantic and textural similarity. Adversarial Loss: A standard GAN loss was implemented, utilizing the discriminator network to guide the generator towards producing more realistic and perceptually convincing images. Loss Weights: The generator loss, G_{Loss}, is defined as a weighted combination of three individual loss components and The equation is: $G_{Loss} = \lambda_{pixel} \cdot L_{pixel} + \lambda_{content} \cdot L_{content} + \lambda_{adv} \cdot L_{adv}$, where we have pixel loss (L_{pixel}), content loss ($L_{content}$), and adversarial loss (L_{adv}). note that more explanation of these equations provided in appendix. Then, the variables of λ_{pixel}, $\lambda_{content}$, and λ_{adv} represent the weights assigned to each loss component. These weights play a crucial role in balancing the influence of each loss function during the training process.
In this implementation and training, the weights were empirically set as follows: $\lambda_{pixel} = 1.0$, $\lambda_{content} = 1.0$ and $\lambda_{adv} = 0.005$. The content loss weight ($\lambda_{content}$) was established as the dominant component, giving priority to perceptual quality, while the adversarial loss weight (λ_{adv}) was significantly lower to provide a subtle influence that encourages the generation of realistic details without destabilizing training. The loss weight of pixels (λ_{pixel}) was set to 1.0.

The subsequent step involved the Optimizer; similar to Stage 1, the Adam optimizer was also utilized in this phase. Following this, the Training Epochs for the USRGAN extended for 200 epochs of training. The ultimate Output and result of this stage was a generator that produced super-resolution images with enhanced perceptual quality, illustrating a trade-off between distortion metrics (PSNR/SSIM) and perceptual realism. To achieve improved training efficiency at the next level, Mixed Precision Training was applied. By leveraging the capabilities of an NVIDIA H100 GPU and the latest versions of PyTorch and Python libraries, we were able to employ bfloat16 mixed precision.

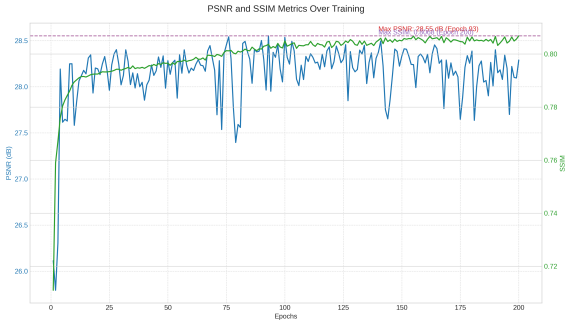

Figure 3. Comparison of PSNR and SSIM metrics, right side as PSNR increase overtime and then stable. right side, increase overtime.

Figure 4. Loss over time, loss function is weighted combination of pixel loss, content loss and adversarial loss

This technique served to reduce memory usage and accelerate computation. Furthermore, we used a Gradient Scaler to ensure stable mixed-precision training. We also developed a new method for Checkpoint Management; the models that exhibited the best performance were saved based on validation metrics. Additionally, we saved the checkpoints in .safetensor format for increased reliability and as a more modern approach. Given the utilization of a high-performance GPU, we also implemented Distributed Training. This implementation included support for multi-GPU training, enabling effective utilization of computational resources.

5. Results and Analysis

This section discusses the results obtained from the SR system and provides an analysis of the quality of the generated images. It is important to note that this work focuses on implementation and inference based on the reference paper [4], and does not include a separate testing phase.

5.1. Training Performance

For this implementation, a Batch Size of 32 images was used for each batch. Subsequently, each Epoch consisted of 74 Iterations, covering our training set of 2,368 images. The Training Time per Epoch ranged from 70 to 85 seconds. The Total Training Time for the complete 200 epochs was

approximately 5 hours. Based on the recorded metrics, the generator loss stabilized within the range of 7.6 to 8.0 after initial fluctuations. The discriminator loss exhibited greater variability, ranging from 0.0001 to 2.69. The Pixel Loss Component demonstrated a decrease from an initial value of 0.46 to approximately 0.08, indicating the model's increasing proficiency in pixel-level reconstruction. The content loss consistently dominated the total loss, maintaining a value of around 98%, which reflects its significant influence on enhancing perceptual quality.

5.2. Quantitative Evaluation

The performance of the super-resolution system was quantitatively evaluated using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), two widely used image quality metrics. there are multiple figues to disscuss about the results but in main text we disscussed about two main figures 3 and 4. others were discussed in apendixes.

PSNR Analysis based on 3, for PSNR, Peak Performance is 28.55 dB at epoch 79. and Secondary Peak at 28.55 dB at epoch 92. Late Training mainly worked to improve performance we can observe A general uptrend after epoch 10, with periodic improvements.

SSIM Analysis Peak Performance is 0.8068 at epoch 200. Trend is Similar to PSNR, the best performance was achieved in the early epochs. The epoch-by-epoch performance showed typical GAN training behavior: Epochs 10-25 we have Rapid improvement in both PSNR and SSIM. Epochs 25-100: Fluctuating metrics with an overall increasing trend. Epochs 100-200: almost stablizing PSNR and still increasing SSIM which means to detect more details in model.

5.3. Qualitative Evaluation

IIt conducted to evaluate the visual quality. This applied to one inference with trained model, more inferences can calculated. The assessment has the following:
in Texture Quality, we canthere is Significant improvement in texture detail compared to bicubic upsampling. for Edge Preservation, there is Sharp edges were observed around structured objects. in Detail Generation, thee are Plausible fine details were generated in complex regions such as hair, foliage, and textured surfaces. and for Artifacts, those exist and Occasional high-frequency artifacts were present in homogeneous regions. also for Color Fidelity, there are Excellent preservation of color information. The model demonstrated a strong ability to generate realistic details consistent with natural image statistics, highlighting the advantage of the GAN-based approach in hallucinating plausible high-frequency information.

Method	PSNR (dB)	SSIM	Visual Quality
Bicubic	21.59	0.6423	Blurry, lacks details
SRCNN	23.85	0.7098	Improved but still soft
RRDB (Ours, Stage 1)	28.55	0.8147	Sharp but lacks texture
USRGAN (Ours, Stage 2)	27.65	0.8568	Best perceptual quality

Table 1. Comparison of Super-Resolution Methods

5.4. Comparative Analysis

The results presented in Table 1 provide comparison of different SR methods. The Bicubic interpolation method, exhibits the lowest PSNR (21.59 dB) and SSIM (0.6423), consistent with its blurry output lacking in fine details. SRCNN [1] shows improvements over Bicubic, achieving higher PSNR (23.85 dB) and SSIM (0.7098); however, its visual quality is still described as soft. Our proposed two-stage approach yields better outcomes in each stage. The RRDB network in Stage 1 achieves the highest PSNR (28.55 dB) and SSIM (0.8147) among the compared methods, indicating its effectiveness in minimizing pixel-wise reconstruction error and preserving structural fidelity. However, It notes that while sharp, the output lacks rich textures. In contrast, USRGAN (Ours, Stage 2), which works as next stage, exhibits a trade-off in terms of PSNR (27.65 dB) and SSIM (0.8568) compared to Stage 1. Its visual quality is rated as the best, shows its success in generating perceptually pleasing HR images with realistic textures.

6. Conclusion

In conclusion, this paper detailed the implementation of a two-stage SR system leveraging RRDBNet followed by GAN architecture. The model focused on achieving high structural fidelity, evidenced by superior PSNR and SSIM metrics along with perceptual quality, resulting in visually compelling HR images with enhanced realism. main benefit of This work is the trade-off between weights of multiple loss functions and the importance of them, there are also other metrics that can be applied here like LPIPS. also, we observed that by changing the weights of every loss function, perceptual quality may change dramatically. As next step in this research, exploring the integration of residual shifting with diffusion models also named ResShift [6] presents a promising path. Diffusion models have shown better capabilities in generating HQ and diverse image contents.

7. Appendix

In this section we want to go deeper through loss functions used for this study and disscuss more about the results.

7.1. Loss function

The training of our SR system employed combination of loss functions to optimize for both structural fidelity and perceptual quality. In the first stage, focusing on the RRDBNet generator, we utilized a pixel-wise loss. The second stage, involving the USRGAN fine-tuning, incorporated loss function comprising pixel loss, content loss, and adversarial loss. The details of each loss component are described below:

7.1.1 Pixel LossL_{pixel}

In both training stages, we used the L1 loss, also known as the Mean Absolute Error (MAE), as a pixel-wise loss function. Given a generated SR image and its corresponding HR ground truth image the L1 loss is calculated as the average absolute difference between their pixel values:

$$L_{pixel}(I_{SR}, I_{HR}) = \frac{1}{N}\sum_{i=1}^{N}|I_{SR}^{(i)} - I_{HR}^{(i)}| \tag{1}$$

where N is the total number of pixels in the image, and $I_{SR}^{(i)}$ and $I_{HR}^{(i)}$ denotes the value of the i-th pixel. This loss function encourages the generator to produce SR images that are pixel-wise close to the HR targets, primarily focusing on structural similarity and aiming to maximize quantitative metrics like PSNR. In the first training stage, this was only loss function, emphasizing the accurate reconstruction of the underlying image structure. In the second stage, while still used, its weight was reduced to allow the network to prioritize perceptual enhancements.

7.1.2 Content Loss$L_{content}$

To improve the perceptual quality of the generated SR images in the second training stage, we employed a content loss based on the feature maps of a pre-trained VGG19 network. This loss measures the high-level feature differences between the generated SR image and the corresponding HR image, encouraging semantic and textural similarity. Specifically, for a given layer l of the VGG19 network, the content loss is defined as the squared Euclidean distance between the feature representations $\phi_l(I_{SR})$ and $\phi_l(I_{HR})$

$$L_{content}(I_{SR}, I_{HR}) = \frac{1}{C_l H_l W_l}\sum_{c=1}^{C_l}\sum_{h=1}^{H_l}\sum_{w=1}^{W_l}(\phi_l(I_{SR})_{c,h,w} - \phi_l(I_{HR})_{c,h,w}) \tag{2}$$

where denotes $\phi(.)$ the feature map extracted by the l-th layer of the VGG19 network, and C_l,H_l and W_l are the number of channels, height, and width of the feature map at that layer, respectively. By minimizing this loss, the generator is encouraged to produce SR images that share similar perceptual characteristics with the real HR images, leading to more visually plausible textures and details. In our implementation, we utilized the feature maps from a high-level layer of the VGG19 network to capture semantic content effectively.

7.1.3 Adversarial Loss (L_{adv})

To further enhance the realism of the generated SR images in the second training stage, we used an adversarial loss within the GAN framework. This loss is based on the ability of a discriminator network (D) to distinguish between real HR images and generated SR images ($G(I_{LR})$). We employed a standard non-saturating GAN loss, where the generator aims to minimize the probability of the discriminator correctly classifying its generated images as fake:

$$L_{adv}(G) = \mathbb{E}_{I_{LR}}[\log(1 - D(G(I_{LR})))] \tag{3}$$

The discriminator, on the other hand, is trained to maximize the probability of correctly classifying both real HR images and generated SR images:

$$L_{adv}(D) = \mathbb{E}_{I_{HR}}[\log(D(I_{HR}))] + \mathbb{E}_{I_{LR}}[\log(1 - D(G(I_{LR})))] \tag{4}$$

By training the generator to fool the discriminator, the adversarial loss encourages the generation of SR images that lie within the distribution of real HR images, resulting in finer and more realistic details that are often missing when relying solely on pixel-wise losses.

7.1.4 Combined Generator Loss(L_{loss})

The total loss function for the generator in the second training stage (G_{Loss}) is a weighted sum of the individual loss components:

$$G_{Loss} = \lambda_{pixel} \cdot L_{pixel} + \lambda_{content} \cdot L_{content} + \lambda_{adv} \cdot L_{adv}, \tag{5}$$

where λ_{pixel}, $\lambda_{content}$, and λ_{adv} are the weights assigned to each loss component. These weights were empirically set to balance the influence of each loss term, allowing us to achieve a desirable trade-off between structural accuracy (driven by pixel loss), perceptual similarity (driven by content loss), and photorealistic detail generation (driven by adversarial loss). In our implementation, the content loss weight was dominant, prioritizing perceptual quality, while the adversarial loss weight was kept relatively low to encourage realistic details without destabilizing the training process.

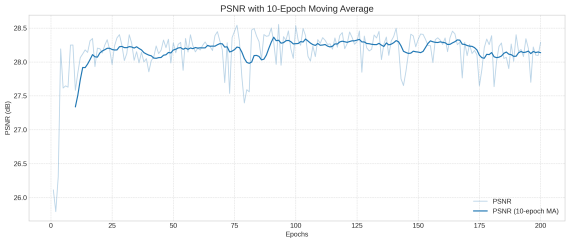

Figure 5. Analyzing PSNR over time and have moving average.

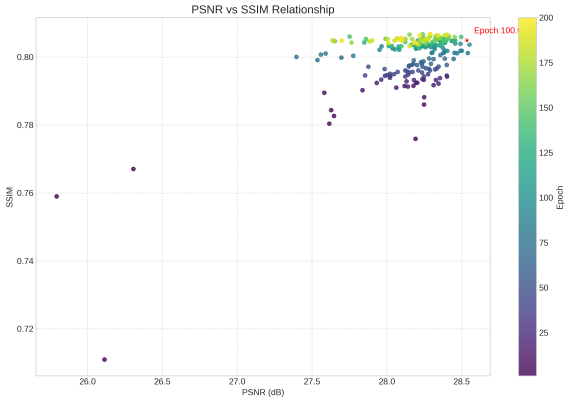

Figure 6. Analyzing relationshop of PSNR and SSIM

7.2. Results

In this section, other than the results that we already presented, we have more figures as comes below, these are outside of the main context of the paper.

as illusterated in figure 5 plot displays the raw PSNR per epoch (blue line) overlaid with a 10-epoch moving average of the PSNR (dark blue line). The moving average smooths out short-term fluctuations in the PSNR metric, making the overall trend clearer and easier to interpret.

as illusterated in figure 3, points represents an epoch. The x-axis is PSNR, the y-axis is SSIM, and the color of the point indicates the epoch number. It demonstrates the relationship between the two main quality metrics. Do they increase together, or is there a trade-off? A red star marks an epoch identified as having a good balance between normalized PSNR and SSIM.

The bar chart illustrates how many epochs it took for the model to reach specific percentages (70%, 80%, 90%, etc.) of its maximum achieved performance for both PSNR and SSIM, based on normalized values. It provides insight into the model's convergence speed. You can see, for example, how quickly the model got close (e.g., 90% or 95%) to its best performance.

and finally, figure **??** and **??** shows the model Inference sample. in this paper main focus was on training. for Inference we used ESRGAN inference method not mentioned

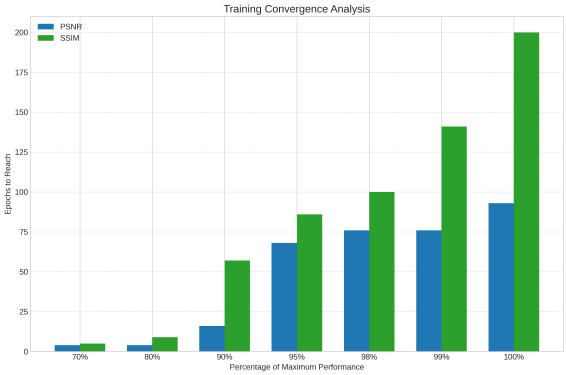

Figure 7. analysis of Training Convergence

Inference section. we have an input then get bicubic downscale(for visualization we wanted to show same scale and used photoshop here!) and rained model Inference, Its like testing the model and see the quality of output.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015. 1, 4

[2] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017. 1

[3] Yawei Li, Kai Zhang, Jingyun Liang, Jiezhang Cao, Ce Liu, Rui Gong, Yulun Zhang, Hao Tang, Yun Liu, Denis Demandolx, Rakesh Ranjan, Radu Timofte, and Luc Van Gool. Lsdir: A large scale dataset for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1775–1787, June 2023. 2

[4] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data, 2021. 1, 2, 3

[5] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018. 1

[6] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting, 2023. 1, 4

[7] Yulun Zhang and Zhang et al. Ntire 2023 challenge on image super-resolution (x4): Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1865–1884, June 2023. 2