

ReactSR: Efficient Real-World Super-Resolution Application in a Single Floppy Disk

Gleb S. Brykin^a and Valeria Efimova^b
ITMO University, Kronverksky Pr, St. Petersburg, Russia

Keywords: Deep Learning, Image Super-Resolution, Image Restoration, Generative Artificial Intelligence, Generative-Adversarial Networks, Vision Transformer, Convolutional Neural Network.

Abstract: Image super-resolution methods are increasingly divided into several groups, setting different goals for themselves, which leads to difficulties when using them in real conditions. While some methods maximize the accuracy of detail reconstruction and minimize the complexity of the model, losing realism, other methods use heavy architectures to achieve realistic images. In this paper, we propose a new class of image super-resolution methods called efficient real super-resolution, which occupies the gap between efficient and real super-resolution methods. The main goal of our work is to show the possibility of creating compact super-resolution models that allow generating realistic images, like SOTA in the field of real super-resolution, requiring only a few parameters and small computing resources. We compare our models with SOTA qualitatively and quantitatively using NIQE and LPIPS image naturalness metrics, getting unambiguous positive results. We also offer a self-contained cross-platform application that generates images comparable to SOTA in terms of realism in an acceptable time, and fits entirely on one 3.5-inch floppy disk.

1 INTRODUCTION

Since the introduction of SRCNN (Dong et al., 2014) in 2014, numerous neural network architectures and training methods have emerged to enhance super-resolution quality and efficiency. In 2016, FSRCNN (Dong et al., 2016) was introduced, significantly speeding up super-resolution and improving image reconstruction accuracy. That same year, the Subpixel Convolution method (Shi et al., 2016) was proposed, enhancing performance and reducing artifacts from transposed convolution used in earlier models.


Subsequent research has focused on complex architectures for precise detail reconstruction (maximizing PSNR/SSIM) and the use of generative adversarial networks (GANs) for photorealistic results. Following the introduction of SRGAN (Ledig et al., 2016), various GANs have been developed, often leading to a divide between classical super-resolution, which emphasizes accuracy, and photorealistic super-resolution, which prioritizes realism over exact reproduction. The work in (Ji et al.,


2020) from 2020 marked a significant advancement in natural super-resolution, distinguishing it from classical methods.

Alongside these developments, there has been a push for efficient architectures suitable for real-time applications on mobile devices. Competitions like NTIRE have emerged, focusing on efficient super-resolution, aiming for accurate image restoration with minimal computational complexity.

Currently, single image super-resolution is categorized into three classes: real, classical, and efficient super-resolution. While each effectively addresses specific tasks, they often fall short in practical applications due to non-photorealistic results or high resource demands. We propose a new class called efficient real super-resolution, which aims to create compact models that process real images with natural distortions and yield photorealistic outcomes.

We tested two models from the NTIRE 2022 (Li et al., 2022) and 2024 (Ren et al., 2024) competitions, training them with state-of-the-art techniques for real super-resolution (Zhang et al., 2021) (Wang et al.,

^a  <https://orcid.org/0009-0006-3063-1948>

^b  <https://orcid.org/0000-0002-5309-2207>

2021) (Liang et al., 2021) (Zhou et al., 2023). The models produced visually appealing images with minimal parameters and computational complexity. As a practical demonstration, we developed cross-platform applications in C# based on these models, achieving acceptable performance on standard laptops without hardware-specific optimizations. The applications fit entirely on a standard 3.5-inch floppy disk and can run on x86, x86-64, or ARM devices with Windows, Linux, or ReactOS, requiring only .NET Framework or Mono.

Our main contributions include:

- The introduction of a new class of image super-resolution methods.
- Enhanced synthetic data generation for efficient model training.
- Improvements to the real super-resolution model training pipeline.
- Demonstration of compact super-resolution models that generate visually pleasing images.
- A production-ready solution that competes with existing alternatives in key metrics.

2 RELATED WORK

The super-resolution problem is categorized into three main groups: real super-resolution, classical super-resolution, and efficient super-resolution. Many studies on neural network architectures propose solutions across these categories. Typically, real-world super-resolution methods incorporate classical super-resolution as an intermediate learning stage.

Super-resolution tasks involve finding an inverse function (Eq. 2) to a downscaling function (Eq. 1), such as bicubic interpolation, which reduces image dimensions while losing information. This irreversibility complicates the upsample function's task, making it ambiguous. The aim of all super-resolution models is to optimize the upsample function based on various metrics. In this context, lr is the low-resolution image, hr is the original high-resolution image, and sr is the high-resolution image produced from lr .

The evaluation of super-resolution functions typically uses formal metrics like PSNR and SSIM, which measure structural similarity, alongside realism metrics such as FID and NIQE, as well as considerations of model complexity.

$$lr = \text{downsample}(hr) \quad (1)$$

$$sr = \text{upsample}(lr) \quad (2)$$

2.1 Classic Super-Resolution

Since (Dong et al., 2014), super-resolution has been framed as finding the inverse of bicubic downsampling. The problem can be viewed as minimizing the distance between sr and hr , commonly using mean squared or absolute error. Classical methods focus on accurately restoring the original image, utilizing PSNR and SSIM for evaluation.

Although some architectures emerged post-2016 that employed generative adversarial networks (GANs), many continued to prioritize precise detail reconstruction (Lim et al., 2017) (Yu et al., 2018) (Ahn et al., 2018) (Zhang et al., 2018). Typically, real super-resolution networks first learn to reconstruct images before being adapted for GAN training (Ledig et al., 2016) (Wang et al., 2018) (Zhang et al., 2021) (Wang et al., 2021) (Liang et al., 2021) (Zhou et al., 2023). However, these classical methods are resource-intensive, often produce less realistic images, and struggle with artifacts in input images, leading to unsatisfactory outcomes for many real images.

2.2 Real-World Super-Resolution

Models in this category aim to effectively process images with natural distortions, such as JPEG artifacts or noise from smartphone cameras. Classical models struggle with such distortions due to their training on ideal data. A 2020 study (Ji et al., 2020) proposed training on distorted images to better handle real-world cases.

Current real super-resolution models build on classical architectures, leveraging their capacity to predict missing elements while incorporating additional training to address distortions. This approach, often combined with GAN strategies, enhances realism and artifact resistance. Recent works (Zhang et al., 2021) (Wang et al., 2021) suggest generating synthetic low-resolution images with simulated distortions for training. The typical training pipeline involves three stages: training on ideal images, further training to suppress artifacts, and GAN training.

Real super-resolution methods prioritize generating visually appealing images, often assessed by FID and NIQE metrics. While PSNR and SSIM can be used, they may not accurately reflect the quality of photorealistic results, which may not match the original texture.

Although real-world super-resolution methods offer visually pleasing results, their high resource demands limit widespread use, and they cannot easily replace non-neural scaling algorithms on end-user devices.

2.3 Efficient Super-Resolution

Efficient super-resolution methods target real-time performance on devices with limited computing resources. Global research efforts focus on developing architectures that balance parameters, computational complexity, and accuracy. Competitions like NTIRE foster innovation in this area.

These models aim to maximize PSNR and SSIM while minimizing resource usage. They often employ complex architectures that outperform simplified versions of classical models; for instance, SMFANet+ (Zheng et al., 2024) demonstrates superior performance compared to SwinIR-light (Liang et al., 2021).

The learning process for efficient super-resolution models mirrors that of classical models, encountering similar issues with defective images and unrealistic generation. However, an additional loss calculated in frequency space enhances the restoration of high-frequency components, albeit not achieving GAN-level quality.

3 PROPOSED METHOD

Our key hypothesis is that compact super-resolution models that have shown good results on the task of efficient super-resolution can be successfully trained like real super-resolution models and produce images of similar quality. To test this hypothesis, we propose an improved learning pipeline, which includes a modified algorithm for generating synthetic training data and a discriminator pre-training stage, which has shown its effectiveness in the task of coloring black and white images and is called NoGAN (DeOldify, n.d.). As the initial models, we chose MobileSR (Sun et al., 2022) and SMFANet+, which demonstrated significant success at the NTIRE 2022 and 2024 competitions and surpassed many previous methods of classical super-resolution in terms of PSNR and SSIM metrics with significantly fewer parameters and computational complexity.

In the end, to confirm the validity of the ideas outlined in the article, we developed cross-platform desktop applications in C# based on trained models, without using hardware-dependent optimizations

such as SIMD, limiting ourselves only to general principles of improving code efficiency, such as optimizing the processor cache by reordering data in memory and multithreading. The solutions obtained were tested both on a laptop and on a server. The results obtained should confirm the possibility of creating compact production-ready solutions that implement real super-resolution.

3.1 Architecture Overview

As test models in this work, we took MobileSR, which is the winner of the model complexity track at NTIRE 2022, as well as the SMFANet model, which took 2nd and 3rd place in the FLOPs and Parameters sub-track of the NTIRE2024 ESR challenge.

MobileSR contains 176,000 parameters, processing an image with a size of 64x64 pixels is the equivalent of about 8.73 GFlops. MobileSR is a hybrid of residual convolutional network and ViT, providing processing of local and global features. MobileSR consists of an input convolutional layer that extracts features from the input image, 5 hybrid blocks ending with convolution, and an upsampling module based on Subpixel Convolution. The hybrid block consists of a visual transformer module with Window Attention and an Inverted Residual Block, similar to the one presented in MobileNet.

SMFANet is represented by two modifications: SMFANet itself and SMFANet+ with an increased number of channels and blocks. SMFANet+ requires more time for inference and has more parameters compared to SMFANet, however, due to significantly better PSNR/SSIM and the insignificant overall complexity of the model, we use SMFANet+ and further under SMFANet we understand this modification. SMFANet contains about 496,000 parameters, processing an image with a size of 64x64 pixels by this model is the equivalent of 28 GFlops. This model is superior to many CNN-based solutions and ViT-based solutions for PSNR/SSIM, while it contains significantly fewer parameters and has less computational complexity. SMFANet consists of a convolution that extracts low-level features from the input image, 12 basic blocks and an effective upsampling module consisting of one convolutional layer and a PixelShuffle that produces the output image. The basic block consists of a self-modulation feature aggregation module and a partial convolution-based feed-forward network. These blocks provide more efficient processing of global and local context than solutions based on vision transformers.

The choice of two fundamentally different architectures (ViT-based and CNN-based) allows for

a more objective assessment of the results obtained, minimizing their dependence on a random architecture choice.

3.2 Data Generation Pipeline

The works (Zhang et al., 2021) (Wang et al., 2021) present a simple and effective way to train an artifact-resistant model. The method consists in generating synthetic low-resolution images containing simulated distortions similar to real ones. Among the basic degradations are compression artifacts, scaling artifacts by various popular algorithms such as bilinear and bicubic interpolation, noise from normal and Poisson distributions, Gaussian blur. In the previous works, these distortions are arranged in sequence in accordance with various patterns. We noticed that the level of artifacts generated by these image generation methods is excessive and the generated images in most cases are unrealistic. For example, the quality parameter of the JPEG codec is randomly selected from the range of 10..95%, which means that a significant part of the images will be overcompressed. Since in reality such a strong compression is practically not found, we suggest using values from the range of 75..99%. In addition, we use fewer duplicate distortions to bring the generated images closer to the real ones. Early synthetic image generation strategies make it possible to train a neural network to handle highly noisy and clamped images well, however, we noticed that such a model would work poorly with high-quality images, removing important details, mistaking them for artifacts. We propose a simple, Real-ESRGAN-like synthetic image generation pipeline (Fig. 1).

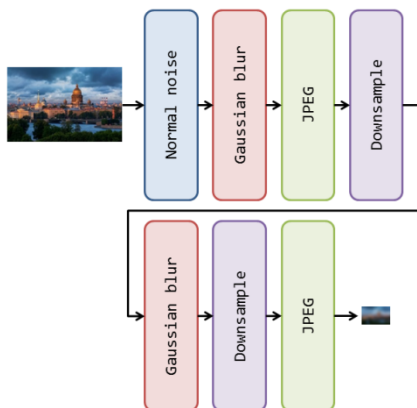


Figure 1: Our synthetic image generation pipeline.

Adding normal noise involves adding random values from the normal distribution multiplied by the w value to each color component of each pixel in the

image. The w value is selected from a uniform distribution in the range $[0, 0.1]$ for each image.

Adding Gaussian blur involves applying convolution with the Gaussian kernel to the image. The kernel size is randomly selected from the set $\{1; 3; 5\}$, the sigma variance parameter is selected from a uniform distribution and lies in the range $[0.001, 0.1]$.

JPEG compression artifacts are added with a 50% probability. When adding artifacts, a compression ratio in the range of $[75, 99]$ % is selected from a uniform distribution.

Downsampling is performed using one of the following methods: nearest neighbor, bilinear interpolation, bicubic interpolation, area. A specific method is randomly selected with equal probability for each of the images each time this operation is performed.

When developing the synthetic data generation pipeline, we tried to rely on distortions that appear on images as a result of natural manipulations with them. Next, we show that for images that initially have acceptable quality, a compact neural network trained by our method can show a better result than SwinIR-large trained on the original pipeline.

3.3 Training Pipeline

The model training pipeline is generally similar to that used in SOTA. However, we suggest adding an additional stage - **discriminator pre-training**. This approach, called NoGAN, was first introduced at DeOldify, where it showed the highest efficiency in training a neural network for coloring black and white photos. We adapt this approach to the super-resolution problem and show (Fig. 2) that such a solution contributes to better learning dynamics at the GAN stage, significantly speeding up the process of transferring knowledge from the discriminator to the generator.



Figure 2: Effect of the discriminator pretraining.

In total, our learning pipeline consists of 4 steps:

- 1) Training the model in accordance with effective super-resolution techniques.
- 2) Retraining the model for the correct processing of synthetic distorted images generated by our method.
- 3) Training the discriminator separately from the generator as an independent neural network.
- 4) Simultaneous generator and the discriminator training (GAN).

We use FFTLoss at all stages of training, with the exception of 4, which uses Focal Frequency Loss (Jiang et al., 2020). Schematically, our learning pipeline is shown in Figure 3.

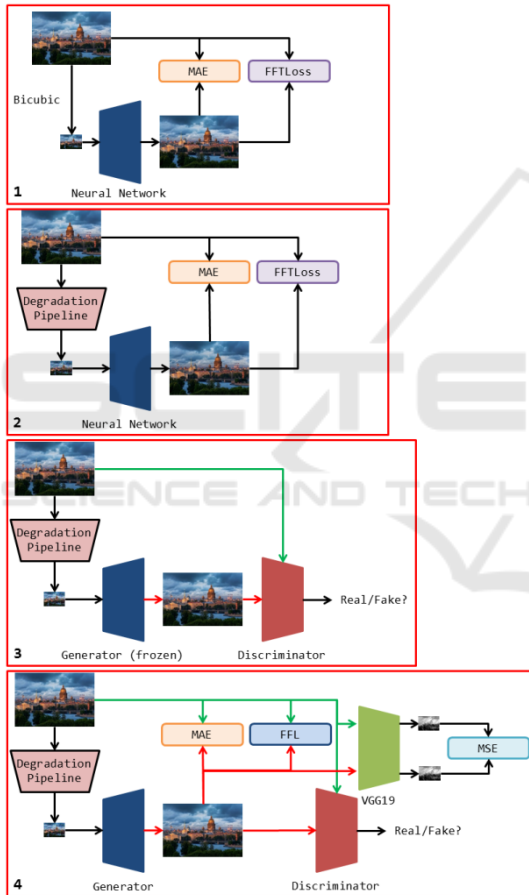


Figure 3: Our training pipeline.

We use the DIV2K dataset for all experiments. The first stage of training in accordance with our pipeline has been omitted, since the pre-trained models of efficient super-resolution provided by the authors are used.

In the second stage, we used the Adam optimizer with an initial learning rate of 5×10^{-4} , $\beta_1=0.9$ and $\beta_2=0.999$. We do 32,000 iterations, halving the

learning rate every 8000 iterations. We set the MAE loss weight to 1 and the FFTLoss weight to 0.1, respectively.

At the 3rd stage of training, we use the Adam optimizer with an initial learning rate of 1×10^{-4} , halved every 8000 iterations. In total, we do 32,000 iterations. We use the RaGAN loss function (Jolicoeur-Martineau, 2018) to train the discriminator. We use the architecture proposed in the work (Wang et al., 2021).

At the last stage of training, we use the Adam optimizer with an initial learning rate of 1×10^{-4} for the generator and discriminator, which is halved every 64,000 iterations. We set the weight for the MAE loss to 1, for the Perceptual Loss the weight is 1, for the Adversarial Loss the weight is 0.1. The α parameter of the Focal Frequency Loss is set to 1, which is optimal for most tasks. Perceptual Loss in our work is represented by the MSE loss in the feature space of the conv4_4 layer of the VGG19 pre-trained neural network.

In all stages, we use a patch size of 16 and a patch size of 128. To speed up I/O, we pre-extract overlapping patches of slightly larger size from the original DIV2K images and save them to disk. In addition to generating synthetic data, we also use general augmentations such as horizontal and vertical flips, as well as patch random cut.

3.4 Inference Framework

Production-ready desktop applications were developed on the basis of the models trained in this work. Since the neural network architectures used in this work are characterized by extremely low computational complexity, we consider it acceptable to implement inference without using hardware-dependent optimizations. By doing this, we aim to simplify the migration of our application to various platforms, simplify deployment, and also ensure the compactness and portability of the solution.

We target our applications to .NET Framework and Mono platforms, using only the cross-platform capabilities of the standard library to provide cross-platform functionality at the binary level, following (Brykin, 2022). We implement all the application code in pure C#, without P/Invoke and third-party libraries. We use common optimizations applicable to various microprocessor architectures, such as optimizing processor cache utilization by redistributing data in RAM when performing convolutions and multithreading. Multithreading is implemented through TPL, which is an integral part of BCL. The model parameters are stored in raw

binary form as a sequence of half-precision floating point values (float16). We found that converting the parameters of the pre-trained model from single to half floating point precision, even without additional training in float16, does not lead to noticeable distortions in the generated images. The correctness of matching parameters in the file and in the model is ensured by a hard-coded sequence of reading values from the file into the internal arrays of the model. To implement the graphical interface, we used Windows Forms, since this technology is supported by various versions of the Windows operating system and ReactOS via .NET Framework, as well as on Linux family operating systems via Mono.

4 EXPERIMENTAL RESULTS

4.1 Implementation Details

We perform experiments and model training using the PyTorch library on a server with NVIDIA Tesla P40 and 2x Intel Xeon 2673v4. We write distributed applications based on our models in C#, focusing them on .NET Framework and Mono platforms and Windows, Linux and ReactOS operating systems. The applications were tested on a laptop with an Intel(R) Core(TM) i5-6300HQ CPU and 32 GB of RAM running the Windows 7 Ultimate x86-64 operating system, as well as on Windows XP x86, Windows Vista x86-64, Windows 8 x86-64, Windows 8.1 x86-64, Windows 10 x86-64, Windows 11 x86-64, ReactOS 0.4.14 x86, Linux Mint 21.3 x86-x64, Ubuntu Desktop 24.04 LTS x86-64, Alt Linux 10.2 Workstation x86-64, Astra Linux SE 1.7 x86-64, Red OS 8 x86-64 and Debian 12.5 x86-64 operating systems, running in the VirtualBox virtualization environment. In addition, testing was conducted on a MacBook Air 2012 with an Intel(R) Core(TM) i7-3667U CPU and 8 GB of RAM running Windows 7 Ultimate x86-64 and on a server with 2x Intel Xeon 2673v4 running Windows 7 Ultimate x86-64.

4.2 Comparisons with SOTA

We qualitatively and quantitatively compare our trained models, which we call Real-MobileSR and Real-SMFANet, respectively, with their original versions, as well as with SwinIR-large, Real-ESRGAN and BSRGAN. For the test, we selected images 0806 and 0809 from the DIV2K test set. The results of numerical comparison with PSNR, SSIM, NIQE and LPIPS metrics are presented in Tables 1 and 2.

Table 1: Comparison of methods on 0806x4.png.

Method/Metric	PSNR	SSIM	NIQE	LPIPS
Real-MobileSR	24.42	0.79	2.21	0.16
MobileSR	23.78	0.79	4.48	0.29
Real-SMFANet	24.85	0.80	2.70	0.13
SMFANet	23.82	0.79	4.60	0.30
BSRGAN	25.26	0.79	2.98	0.15
Real-ESRGAN	24.25	0.77	2.15	0.15
SwinIR-large	24.56	0.80	2.47	0.13

Table 2: Comparison of methods on 0809x4.png.

Method/Metric	PSNR	SSIM	NIQE	LPIPS
Real-MobileSR	26.27	0.72	2.52	0.23
MobileSR	27.01	0.77	5.62	0.32
Real-SMFANet	26.90	0.75	3.42	0.20
SMFANet	26.73	0.77	5.35	0.32
BSRGAN	27.15	0.72	3.68	0.24
Real-ESRGAN	25.97	0.71	3.52	0.25
SwinIR-large	26.26	0.73	3.50	0.21

From Tables 1 and 2, it can be concluded that GAN-based methods are unambiguously superior to non-GAN methods in metrics evaluating the naturalness and visual quality of images (NIQE and LPIPS). From Table 2, we can conclude that our Real-SMFANet in some cases surpasses BSRGAN, Real-ESRGAN and SwinIR-large in NIQE and LPIPS metrics, which are considered SOTA in the field of real super-resolution. In Table 3, we offer a comparison in terms of computational complexity and number of parameters.

Table 3: Comparison of Flops and Params.

Method/Metric	Params (M)	Complexity (GFlops)
Real-MobileSR	0.176	8.73
MobileSR	0.176	8.73
Real-SMFANet	0.496	28
SMFANet	0.496	28
BSRGAN	~8	>50
Real-ESRGAN	~8	>50
SwinIR-large	~17	>100

A qualitative comparison of the methods is presented in Figures 4 and 5. We draw attention to the fact that the visual assessment of the images generated by various models confirms the quantitative assessment given in Tables 1 and 2.

In Figures 4 and 5, we present a comparison of Real-MobileSR, MobileSR, Real-SMFANet, SMFANet, BSRGAN, Real-ESRGAN, SwinIR-large on texture-rich image areas. Using these examples, we can see that methods trained not as real super-resolution sharpen images without increasing detail, while real super-resolution methods do not give

unnaturally sharp generations, but produce many textures, thanks to which images are perceived naturally.

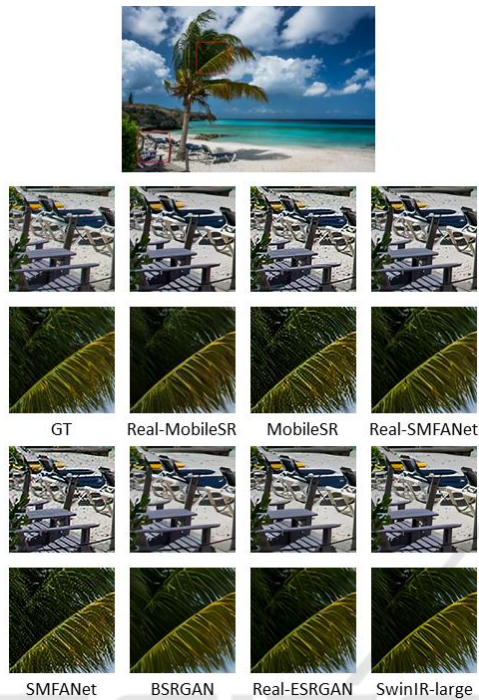


Figure 4: Qualitative comparison on 0806 image from DIV2K.



Figure 5: Qualitative comparison on 0809 image from DIV2K.

4.3 Inference Comparisons

Today, there are several well-known desktop applications designed for image super-resolution. The most famous of them are PhotoZoom Pro, Topaz Gigapixel AI, Waifu2x and Real-ESRGAN. These applications are aimed at the most realistic image restoration, as well as the applications proposed in this work. Due to the fact that PhotoZoom Pro and Topaz Gigapixel AI are proprietary software, it is difficult to unambiguously judge the underlying methods. It is known that PhotoZoom Pro has not used neural algorithms until recently, and Topaz Gigapixel AI obviously follows the trend of real super-resolution in its developments. However, we can compare software products by the disk space occupied by the program modules, as well as by the size of the installation distributions.

An analysis of existing analogues allowed us to conclude that the minimum disk space required to install the software is 74 MB for Real-ESRGAN on Windows. Topaz Gigapixel requires more than 1 GB to automatically download models, the current version of PhotoZoom Pro takes up 212 MB of disk space.

The applications we have developed in this work require 548 KB and 1.13 MB, respectively, which is several orders of magnitude less. Our programs are so compact that they can be written to a single standard 3.5-inch floppy disk. Additionally, we can mention a wide range of operating systems and microprocessor architectures supported by our applications.

5 CONCLUSION

The new trend of image super-resolution proposed in this paper may become a new trend in the field of image restoration. Efficient real super-resolution combines the advantages of the methods of two actively developing areas of super-resolution and allows solving practical problems. In this paper, we proposed a pipeline for training models of effective real super-resolution, trained two different models with its help and evaluated them qualitatively and quantitatively, confirming the validity of our hypothesis. We suggest that researchers working on effective super-resolution conduct additional research on their architectures for the possibility of their application to real super-resolution task. The developed desktop applications are available on GitHub and can be freely tested by the community: <https://github.com/ColorfulSoft/ReactSR>

ACKNOWLEDGEMENTS

The research was supported by the ITMO University, project 623097 “Development of libraries containing perspective machine learning methods”.

REFERENCES

- Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Image super-resolution using deep convolutional networks. *arXiv e-print*. <https://arxiv.org/abs/1501.00092>
- Dong, C., Loy, C. C., & Tang, X. (2016). Accelerating the super-resolution convolutional neural network. *arXiv e-print*. <https://arxiv.org/abs/1608.00367>
- Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *arXiv e-print*. <https://arxiv.org/abs/1609.05158>
- Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2016). Photo-realistic single image super-resolution using a generative adversarial network. *arXiv e-print*. <https://arxiv.org/abs/1609.04802>
- Ji, X., Cao, Y., Tai, Y., Wang, C., Li, J., & Huang, F. (2020). Real-world super-resolution via kernel estimation and noise injection. *arXiv e-print*. <https://arxiv.org/abs/2005.01996>
- Li, Y., Zhang, K., Timofte, R., Van Gool, L., & others. (2022). NTIRE 2022 challenge on efficient super-resolution: Methods and results. *arXiv e-print*. <https://arxiv.org/abs/2205.05675>
- Ren, B., Li, Y., Mehta, N., Timofte, R., & others. (2024). The ninth NTIRE 2024 efficient super-resolution challenge report. *arXiv e-print*. <https://arxiv.org/abs/2404.10343>
- Lim, B., Son, S., Kim, H., Nah, S., & Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution. *arXiv e-print*. <https://arxiv.org/abs/1707.02921>
- Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., & Huang, T. (2018). Wide activation for efficient and accurate image super-resolution. *arXiv e-print*. <https://arxiv.org/abs/1808.08718>
- Ahn, N., Kang, B., & Sohn, K.-A. (2018). Fast, accurate, and lightweight super-resolution with cascading residual network. *arXiv e-print*. <https://arxiv.org/abs/1803.08664>
- Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., & Fu, Y. (2018). Image super-resolution using very deep residual channel attention networks. *arXiv e-print*. <https://arxiv.org/abs/1807.02758>
- Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., & Tang, X. (2018). ESRGAN: Enhanced super-resolution generative adversarial networks. *arXiv e-print*. <https://arxiv.org/abs/1809.00219>
- Zhang, K., Liang, J., Van Gool, L., & Timofte, R. (2021). Designing a practical degradation model for deep blind image super-resolution. *arXiv e-print*. <https://arxiv.org/abs/2103.14006>
- Wang, X., Xie, L., Dong, C., & Shan, Y. (2021). Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. *arXiv e-print*. <https://arxiv.org/abs/2107.10833>
- Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., & Timofte, R. (2021). SwinIR: Image restoration using Swin transformer. *arXiv e-print*. <https://arxiv.org/abs/2108.10257>
- Zhou, Y., Li, Z., Guo, C.-L., Bai, S., Cheng, M.-M., & Hou, Q. (2023). SRFormer: Permuted self-attention for single image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- Zheng, M., Sun, L., Dong, J., & Pan, J. (2024). SMFANet: A lightweight self-modulation feature aggregation network for efficient image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
- DeOldify. (n.d.). *DeOldify homepage*. GitHub. <https://github.com/jantic/DeOldify>
- Sun, L., Pan, J., & Zhu, Y. (2022). MobileSR: A mobile-friendly transformer for efficient image super-resolution. In *Proceedings of the NTIRE 2022 Workshop on Image Super-Resolution*.
- Jiang, L., Dai, B., Wu, W., & Loy, C. C. (2020). Focal frequency loss for image reconstruction and synthesis. *arXiv e-print*. <https://arxiv.org/abs/2012.12821>
- Jolicœur-Martineau, A. (2018). The relativistic discriminator: A key element missing from standard GAN. *arXiv e-print*. <https://arxiv.org/abs/1807.00734>
- Brykin, G. S. (2022). The System.AI project: Fully managed cross-platform machine learning and data analysis stack for .NET ecosystem. In *Trudy Instituta sistemnogo analiza Rossiyskoy akademii nauk (ISARAN) (Proceedings of the Institute for Systems Analysis Russian Academy of Sciences (ISA RAS))*, pp. 64-72. <https://doi.org/10.14357/20790279230108>