VisionRefine: High-Resolution Image Recovery

Jiahe Huang, Shixuan Liu, Xuejun Zhang, Jingjing Zhu, Shuangyu Lei
University of Michigan, Ann Arbor, MI, USA
{chloehjh, shixuanl, xuejunzh, zhujudy, sylei}@umich.edu

Abstract

In our study, we tested three models to enhance image resolution: SRCNN with structural modifications (2% PSNR increase), two guided diffusion models (1% PSNR increase and improved details), and a CodeFormer algorithm with facial/non-facial area balancing.

1. Introduction

Super-resolution (SR) is an important area in image processing that connects theoretical research and practical applications. It involves converting low-resolution (LR) images into high-resolution (HR) counterparts, which is particularly useful when capturing HR images is difficult or expensive, but LR images are readily available.

Our research focuses on solving the problem of upscaling LR images to HR through the use of advanced computational methods. Our main goal is to explore how AI models can be effectively employed to upscale LR images to HR while preserving or even improving the image quality.

We propose three potential solutions to the image super-resolution problems. Previous research has shown that simple CNN architecture can be efficiently used in improving image fidelity [3], and has decent generalization ability among different distributions of images. Apart from these classic models, diffusion models [2] as well as transformer based encoder-decoder models [8] can also be applied for super resolution tasks on a particular dataset, where only one or a few classes of objects are concerned. While these models can achieve SOTA performances on specific tasks, their robustness to distribution shift can be compromised.

Our work in super-resolution has immense practical implications in various fields, such as satellite imaging, medical imaging, and surveillance. Improved clarity of visual data can greatly enhance the output and usability of these fields. Furthermore, by advancing SR techniques, we contribute to the broader AI and computer vision community, providing them with tools and methodologies that can be adapted and improved.

In summary, our contributions are as follows. (1) We recover the image resolution through deep convolutional networks (Section 3.2), diffusion models (Section 3.3.1, 3.3.2), and transformers (Section 3.4) by reproducing and enhancing SOTA works. (2) We compare the outputs of different methods to understand the limitations of each work.

2. Background

The deep convolutional network is a widely used method for improving image quality. In their work, Dong et al. [3] introduced a fully convolutional network that learns the mapping from LR to HR images. This methodology bridges the gap between deep-learning-based SR techniques and traditional sparse-coding-based SR methods, leveraging the strengths of deep-learning architectures to improve upon the limitations of earlier approaches.

Furthermore, generative models are gaining prominence in addressing similar challenges. Diffusion Posterior Sampling (DPS) in [1] employs a guided sampling process within diffusion models to steer the output toward desired characteristics, such as enhanced resolution. Similarly, InfEdit in [7] presents a more stable and effective approach for achieving inversion-based image editing, showcasing advancements in generative modeling techniques.

Transformers represent another innovative architecture within the realm of image editing. In their study, Zhou et al. [8] introduce CodeFormer, a transformer-based methodology that significantly enhances resolution recovery. This approach integrates a discrete codebook and a decoder, complemented by a controllable feature transformation module. This module effectively regulates the flow of information from the LQ encoder to the decoder, thereby improving overall image resolution.

Readers of our research are highly recommended to have deep learning backgrounds, especially with convolutional neural networks. A brief understanding of the sampling algorithm of diffusion models and the transformer encoder-decoder architecture is also required.

3. Methodology

3.1. LR-HR Image Pair Generation

To obtain input and ground truth pairs, we downsample HR images to obtain LR inputs. The `torch` library is used throughout the project.

3.2. Deep Convolutional network(SRCNN Model)

We leverage the Super-Resolution Convolutional Neural Network (SRCNN) model [3], which has demonstrated remarkable performance in upscaling low-resolution images to high-resolution counterparts. We reuse their overall idea and modify the detailed implementation to improve.

The SRCNN architecture comprises three main layers: patch extraction, non-linear mapping, and reconstruction. As the low-resolution input is first upscale to the desired size using bicubic interpolation before inputting to SRCNN network. Thus, we denote the ground truth high-resolution image as $\mathbf{X}$, the Bicubic upsampled version of low-resolution image as $\mathbf{Y}$.

1. **Patch Extraction Layer**: The first layer perform a standard conv with Relu to get F1($\mathbf{Y}$).

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1) \tag{1}$$

Input image $\mathbf{Y}$ are divided into overlapping patches with a predefined stride. These patches serve as input to the subsequent layers of the network. The size of filters W_1 is $c \times f_1 \times f_1 \times n_1$where c is number of channels of the image, f_1 is the filter size, and n_1 is the number of filters. B_1 is the n1-dimensional bias vector. In our model, $f_1 = 9$, $n_1 = 64$.

2. **Non-linear Mapping Layer**: A non-linear mapping is performed in this layer.

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2) \tag{2}$$

This convolutional layer learn complex mappings between LR and HR image patches. The depth and capacity of the network are determined by the number of filters and their spatial dimensions. In out model, we set $W_2 = n_1 \times f_2 \times f_2 \times n_2 = 64 \times 5 \times 5 \times 32$.

3. **Reconstruction Layer**: The final convolutional layer aggregates the high-dimensional feature maps produced by the previous layers and reconstructs the high-resolution image patches.

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3 \tag{3}$$

No activation function is applied in this layer to preserve the full range of pixel values. In this layer, $W_3 = n_2 \times f_3 \times f_3 \times c = 32 \times 5 \times 5 \times c$.

3.3. Guided Diffusion Models

The diffusion models can be guided to refine low-resolution images towards a higher fidelity output through iterative denoising steps. In this section, we present two strategies, using the diffusion model as well as its variant, consistency model to perform image super resolution under the guidance and conditioning of low resolution images.

3.3.1 Super Resolution Based on Classifier Guided DDPM Sampling

The ODE formulation of diffusion probability is Eq. 4.

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}; \sigma(t)\big)\mathrm{d}t \tag{4}$$

We basically follow the pipeline put forward in [2] to perform super-resolution. The super-resolution diffusion model is pre-trained to sample high-resolution images conditioned on low-resolution input. The low-resolution image is first upscaled with bilinear interpolation and concatenated to the input as 3 additional channels.

The details of the sampling algorithms are shown in Alg. 1 and Alg. 2, and the guided sampling algorithm is the same as the one in [2]. Different from the original pipeline, we also combine the classifier-guided diffusion sampling technique during the upsampling process.

Algorithm 1: Classifier guided diffusion sampling

Data: Class label y, gradient scale s,
diffusion model weights ϵ_θ,
classifier weights ϕ
Result: Generated image x_0
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for $t \leftarrow T$ **to** 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from
   $\mathcal{N}(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end

Algorithm 2: Classifier guided DDIM sampling

Data: Class label y, gradient scale s,
diffusion model weights ϵ_θ,
classifier weights ϕ
Result: Generated image x_0
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for $t \leftarrow T$ **to** 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end

3.3.2 EDM Based Diffusion Posterior Sampling

Furthermore, we combine the EDM sampler [5] with DPS [1] to obtain the new guided sampling algorithm Alg. 3

Algorithm 3: EDM based DPS

Data: StochasticSampler $D_\theta(\boldsymbol{x};\sigma)$,
$t_{i\in\{0,\dots,N\}}$, $\gamma_{i\in\{0,\dots,N-1\}}$, S_{noise}
sample $\boldsymbol{x}_0 \sim \mathcal{N}\left(\mathbf{0},\ t_0^2\mathbf{I}\right)$
for $i \in \{0,\dots,N-1\}$ **do**
  sample $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0},\ S_{noise}^2\mathbf{I}\right)$
  $\hat{t}_i \leftarrow t_i + \gamma_i t_i$
  $\hat{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \sqrt{\hat{t}_i^2 - t_i^2}\boldsymbol{\epsilon}_i$
  $\boldsymbol{d}_i \leftarrow \left(\hat{\boldsymbol{x}}_i - D_\theta\left(\hat{\boldsymbol{x}}_i;\hat{t}_i\right)\right)/\hat{t}_i$
  $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\boldsymbol{d}_i$
  $\hat{\boldsymbol{x}}_N \leftarrow D_\theta\left(\hat{\boldsymbol{x}}_i;\hat{t}_i\right)$
  if $t_{i+1} \neq 0$
    $\boldsymbol{d}'_i \leftarrow \left(\boldsymbol{x}_{i+1} - D_\theta(\boldsymbol{x}_{i+1};t_{i+1})\right)/t_{i+1}$
    $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\left(\frac{1}{2}\boldsymbol{d}_i + \frac{1}{2}\boldsymbol{d}'_i\right)$
    $\hat{\boldsymbol{x}}_N \leftarrow D_\theta(\boldsymbol{x}_{i+1};t_{i+1})$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_{i+1} - \zeta_i\nabla_{x_i}\|y - \mathcal{A}(\hat{\boldsymbol{x}}_N)\|_2^2$
end
return $\boldsymbol{x}_N$

where $\mathcal{A}(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^n$ is resizer from [1], a linear forward measurement operator. The general forward model is Eq. 5.

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_N) + \boldsymbol{n}, \quad \boldsymbol{y},\boldsymbol{n} \in \mathbb{R}^n, \boldsymbol{x} \in \mathbb{R}^d \tag{5}$$

The stochastic sampler can be explained as Eq. 6 according to the EDM paper [5].

$$\begin{aligned}\mathrm{d}\boldsymbol{x}_\pm = &\underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t))\mathrm{d}t}_{\text{probability flow ODE (Eq. 4)}} \\ &\pm \underbrace{\beta(t)\sigma(t)^2\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t))\mathrm{d}t}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)\ \mathrm{d}\omega_t}_{\text{noise injection}}\end{aligned} \tag{6}$$

Combined with DPS, our guided EDM method provides a new sampling way as Eq. 7 where $\boldsymbol{y}$ is a partial measurement derived from $\boldsymbol{x}$ and $p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})/p(\boldsymbol{y})$.

$$\begin{aligned}\mathrm{d}\boldsymbol{x}_\pm = &-\dot{\sigma}(t)\sigma(t)[\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t)) + \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)]\mathrm{d}t \\ &\pm \beta(t)\sigma(t)^2[\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t)) + \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{y}|\boldsymbol{x}_t)]\mathrm{d}t \\ &+ \sqrt{2\beta(t)}\sigma(t)\ \mathrm{d}\omega_t\end{aligned} \tag{7}$$

3.4. Transformer Model

We enhanced an existing deep learning algorithm, "CodeFormer," which is designed to restore high-quality facial images from severely degraded low-quality images [8]. Originally based on a transformer model to predict and restore facial details, we introduced additional steps of face masking and image sharpening. These new steps aim to protect facial regions within the image from over-processing,

Algorithm 4: Streamlined CodeFormer with Face Masking and Sharpening

Data: Low-quality image I_L
Result: Enhanced and restored image I_R
Initialize Encoder E, Decoder D, Transformer T, CFT module
for *each training image* **do**
  Encode image to features F
  Quantize features to get code tokens
  Decode to reconstruct image
  Compute and apply gradients
end
for *each* I_L **do**
  Detect faces and calculate masks
  Apply Gaussian blur outside masks
  Predict codes with Transformer T
  Decode codes to get preliminary I_R
  Apply CFT to adjust features
  Sharpen non-masked areas of I_R
end

while applying Gaussian blur and sharpening to the background and non-facial areas to enhance the overall visual quality and clarity of the image.The detailed algorithm is shown in Alg. 4.

1. **Preprocessing and Core Processing with CodeFormer**: The algorithm first encodes a low-quality image into a feature space and quantizes these features into discrete codes. Using a Transformer, the algorithm predicts the sequence of codes that represents the degraded input most effectively. The codes are then decoded back into an image, reconstructing the preliminary high-quality features, primarily focusing on the facial details.
2. **Masking and Enhancement**: After the initial restoration using CodeFormer, the algorithm identifies facial regions in the restored image. These regions are masked to protect them from the subsequent enhancement steps. Gaussian blur is applied to the non-facial areas of the restored image to smooth out imperfections and focus the visual clarity on the facial features. For enhancing clarity and detail in the non-masked, non-facial areas of the image, a sharpening filter is applied using a specific kernel:

$$\text{sharpening kernel} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

This kernel amplifies the edges and fine details, improving the overall visual quality and ensuring that the background complements the restored facial features effectively. The non-masked, non-facial areas of the image are then sharpened to enhance clarity and detail, improving the overall visual quality and ensuring the background complements the

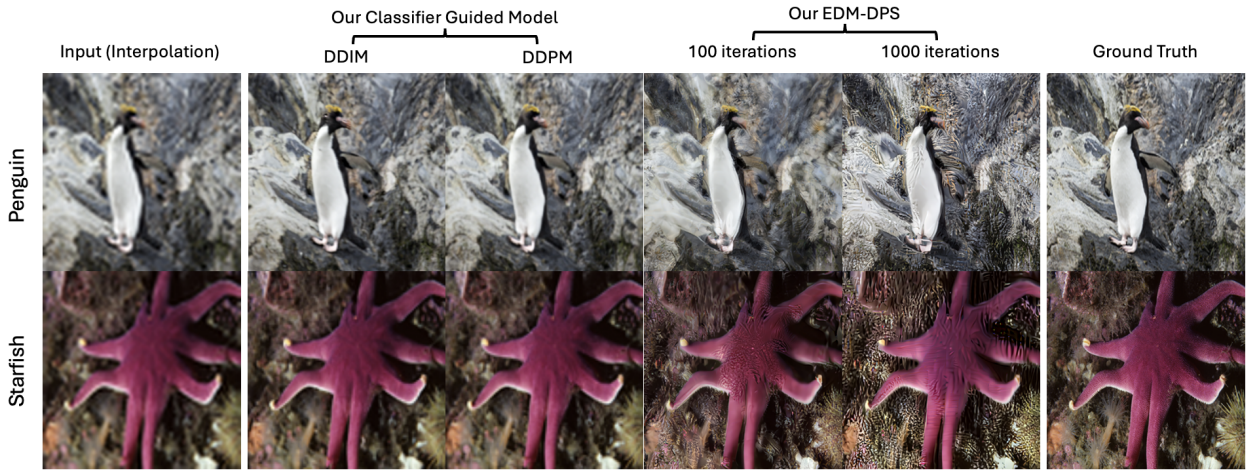

Figure 1. Results on up-sampling penguin and starfish.

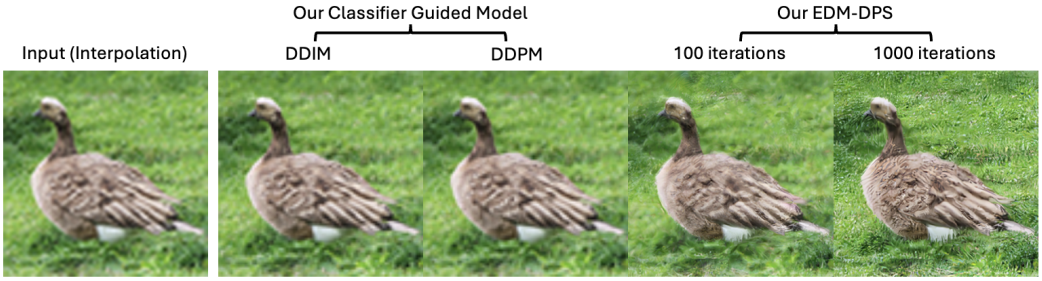

Figure 2. Results on up-sampling randomly-generated duck image.

restored facial features effectively.

4. Results

4.1. SRCNN Model

We evaluate the performance of the Super-Resolution Convolutional Neural Network (SRCNN) model on the DIV2K dataset. Due to computational constraints, we conducted experiments on a subset of the dataset, using two randomly selected images for evaluation. The peak signal-to-noise ratio (PSNR) values of the SRCNN model and baseline interpolation methods are presented in Table 1. The resulting figure is shown in Figure 4 and 6.

Model	Interpolation	SRCNN
penguin	29.00	**29.42**
starfish	30.06	**30.25**

Table 1. Super Resolution Results of SRCNN Model

In order to get a better result, we chose to implement a 9-5-5 architecture for the SRCNN model, with a filter size of 5 in the non-linear mapping layer ($f_2 = 5$), instead of the typical 9-1-5 architecture. This decision was made to capture more complex features and enhance the model's ability to learn intricate details, albeit at the expense of longer training times.

Figure 3. penguin Interpolation result

Figure 4. penguin SRCNN result

Despite the increased computational cost and training time associated with the 9-5-5 architecture, our experimentation indicates that it leads to improved super-resolution performance. The SRCNN model with the 9-5-5 architec-

Figure 5. starfish Interpolation result

Figure 6. starfish SRCNN result

ture consistently outperforms baseline interpolation methods, as demonstrated by the higher PSNR values. Overall, we guarantee a $1-2\%$ improvement in PSNR.

4.2. Guided Diffusion and Consistency Models

We use the guided diffusion model and consistency models to perform zero-shot image super resolution on the DIV2K dataset [4]. We reuse the pre-trained model for DDIM and DDPM in [2] and train our own model for EDM. Detailed experiment is introduced in Appendix C. Due to computing limitations, we only experimented on two images from the dataset. The PSNR values of different models for two randomly sampled images are listed in Table 2.

Model	Interpolation	DDPM	DDIM	EDM
penguin	29.00	**29.20**	29.01	28.90
starfish	30.06	**30.49**	30.46	29.31

Table 2. Super Resolution Results of Guided Diffusion and Consistency Models

We pick the raw bilinear interpolation output as a baseline for reference. The results here demonstrate a trade-off between the efficiency and quality of sampling. Although the quality of samples is generally lower for the DDIM sampler or consistency model architecture, we can also notably reduce the number of time steps, from 1000 steps to 250 steps or even 100 steps. While the improvement is limited in terms of the absolute difference of PSNR that we have only about 1% improvement, we can indeed observe the true enhancement from Fig. 1. The low PSNRs of our guided diffusion models are possibly due to computing limitations and are further discussed in Appendix C.3. One notable feature of the guided EDM model is that it enhances the details while not significantly improving the overall resolution, and the guided DDPM and guided DDIM models focus more on the latter. Thus, the low PSNR of our guided EDM is likely due to this feature as the ground truth only improves the overall resolution.

When testing with our custom images, we also find that for some other situations, EDM-DPS performs the best. In Fig. 2 we randomly generate a low-resolution duck image using a pre-trained model in [2] and the guided EDM model improves the resolution significantly.

4.3. Transformer Model

In this section, we present the results of our enhanced CodeFormer algorithm. We evaluated the performance using two commonly accepted metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).
Our enhanced algorithm achieved a PSNR of 36.32 dB, indicating high-quality restoration well above typical ranges observed in standard scenarios (20 to 40 dB). This high PSNR suggests that the restored image maintains a high fidelity with minimal distortion. Additionally, the SSIM value reached 0.8975, nearing the perfect score of 1, which confirms that the structural integrity and texture details of the restored image closely resemble those of the original. This demonstrates effective preservation of crucial facial features, enhancing perceptual quality. The visual results are presented in Figure 7, 8, 9, 10

Figure 7. Original Image

Figure 8. Enhanced Image

Figure 9. Original Image

Figure 10. Enhanced Image

5. Conclusion

Based on our investigation into image super-resolution using the SRCNN model, we obtained promising results. We adopted a 9-5-5 architecture that strikes a balance between computational efficiency and model performance, resulting in enhanced reconstruction fidelity in comparison to baseline interpolation methods. We also achieved a guaranteed improvement in PSNR of 2%. For the guided diffusion models, we found that the EDM-based DPS model

excels in upsampling detailed features, while the guided DDPM/DDIM models are better suited for improving overall resolution. We also achieved a guaranteed improvement in PSNR of 1%. Regarding the transformer model, we introduced modifications to the CodeFormer algorithm to refine the process of facial image restoration. The implementation of targeted face masking and selective sharpening helped better balance the treatment of facial and non-facial areas, reflecting a thoughtful approach to addressing specific challenges in image restoration.

In the future, more stable diffusion models on larger datasets and higher resolution (*e.g.* 512×512) should be trained. Further study on the behavior of the EDM-DPS model on its sharpening feature will be conducted in the future. Our team is committed to further developing the CodeFormer algorithm to address specific challenges encountered in facial image restoration. Future efforts will be dedicated to optimizing and refining our unique enhancements, including more targeted face masking and selective sharpening. Additionally, we will explore the integration of new techniques to handle varying levels of degradation more effectively.

References

[1] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023. 1, 2, 3, 6, 7

[2] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. 1, 2, 5, 6, 7

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015. 1, 2, 6

[4] Andrey Ignatov, Radu Timofte, et al. Pirm challenge on perceptual image enhancement on smartphones: report. In *European Conference on Computer Vision (ECCV) Workshops*, January 2019. 5, 7

[5] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. 2, 3, 6, 7

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 7

[7] Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with natural language. 2024. 1

[8] Shangchen Zhou, Kelvin C.K. Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. In *NeurIPS*, 2022. 1, 3, 6, 7

A. Codes

Our codes are available at https://github.com/sx-liu/EECS442SR. Different branches contain different methods.

Data Generation Despite the datasets mentioned in the paper, we also write our own downsampling script to obtain LR images from HR ones.

SRCNN model The SRCNN model code is based on the main structures from the following repositories https://github.com/yjn870/SRCNN-pytorch [3] and fine-tuned by ourselves.

Diffusion models We reuse the main structures from the following repositories https://github.com/openai/guided-diffusion [2], and add our own sampling and conditioning methods to it. For the EDM-based DPS model, we reuse the training codes from [5] to obtain a pre-trained model and we also reuse the resizer in [1] for guidance. The guided sampling codes are based on the non-guided EDM sampler in [5] and are designed and fine-tuned by ourselves.

Transformer model We reuse the main code structure of CodeFormer [8] and enhance it by ourselves as mentioned in Section 3.4.

B. SRCNN Experiment

B.1. Pretrained Model

Our SRCNN model is trained on the T-91 image dataset, which is more suitable for its simplicity and inability to learn high-detail features effectively. The T-91 image dataset can be found at https://www.kaggle.com/datasets/ll01dm/t91-image-dataset. The training is executed locally, utilizing a 1060Ti GPU, and takes approximately 80 minutes.

B.2. Data Preparation

The script prepares training and evaluation datasets for a Super-Resolution Convolutional Neural Network (SRCNN). Training data consists of patches extracted from low and high-resolution images, stored in an HDF5 file.

$$\text{train(args)} = \text{lr\_patches}, \text{hr\_patches} \tag{8}$$

On the other hand, evaluation data consists of entire low and high-resolution images.

$$\text{eval(args)} = \text{lr\_group}, \text{hr\_group} \tag{9}$$

Figure 11. Facial masks visualization

The script accepts command-line arguments for the images directory, output path, patch size, stride, scaling factor, and a flag for evaluation mode.

B.3. Low PSNR Reasons for SRCNN

The PSNR of our SRCNN model are not satisfying in Table 1, which could be due to the following reasons:

1. **Inherent limitations of the SRCNN model.** The SRCNN model is a simpler model and struggles to learn high-detail features effectively. This limits its performance on super-resolution tasks where high-detail features are crucial.

2. **Training on the T-91 image dataset.** The SRCNN model was trained on the T-91 image dataset, which is of lower resolution. While this dataset is more suitable for the SRCNN model, it also limits the ability of the model to generate high-resolution outputs.

C. Guided Diffusion Experiment

C.1. Classifier Guided DDPM Sampling

C.1.1 Pretrained Model

We reuse the $64 \times 64 \rightarrow 256 \times 256$ class-conditional model in [2] and can be directly downloaded at https://openaipublic.blob.core.windows.net/diffusion/jul-2021/64_256_upsampler.pt. The model is trained on the ImageNet dataset [6].

C.1.2 Sampling Setup

For the classifier guidance, shown in Eq. 10, we choose the scaling factor for the classifier gradient to be $s = 4.0$.

$$x_{t-1} \sim \mathcal{N}(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y|x_t), \Sigma) \tag{10}$$

C.2. EDM-based DPS Setup

C.2.1 Pretrained Model

Our model is trained on the DIV2K dataset [4] using EDM trainer [5]. The pre-trained model achieves a training error of around 15% due to the limited training set and can be downloaded at our shared Google drive https://drive.google.com/file/d/1AGy7nSMq9UQgG0wZ6wg4DLWPS2o1vVtT/view?usp=sharing. The model has been trained for 12 hours on 3 A40 GPUs.

C.2.2 Sampling Setup

The forward model for super-resolution is defined as Eq. 11 where $\boldsymbol{L}^f \in \mathbb{R}^{n \times d}$ is the bicubic downsampling block Hankel matrix according to [1].

$$y \sim \mathcal{N}(\boldsymbol{y}|\boldsymbol{L}^f\boldsymbol{x}, \sigma^2\boldsymbol{I}) \tag{11}$$

Moreover, for the selection of ζ_i in Alg. 3, we rewrite the last step as Eq. 12 where $\zeta_i = 40$ works best.

$$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_{i+1} - \zeta_i \nabla_{x_i} \|y - \mathcal{A}(\hat{\boldsymbol{x}}_N)\|_2 \tag{12}$$

C.3. Low PSNR Reasons

The PSNR of our guided diffusion models are not satisfying in Table 2, which is possibly due to the following reasons:

1. **Low resolution of the pre-trained model.** Our pre-trained models have only a resolution of 256×256, which is insufficient for a super-resolution where resolution greater than 1024×1024 is wanted. For higher performance on the super resolution task, fine-tuning on DIV2k or other SR dataset is desirable.

2. **Small size of the training size.** For our EDM model, we use the DIV2K dataset [4] which only contains 800 training images. This is insufficient for diffusion models and may lead to overfitting, making testing error high.

D. Transformer Experiment

D.1. Pretrained Model

We reuse the pre-trained model and dataset of CodeFormer [8].

D.2. Facial Mask Visualization

A sample of the facial mask visualization is as Fig. 11, which is an intermediate output of our transformer method.