General Image Restoration via Conditional Diffusion Probabilistic Model

Dewei Hu

Specific Aims: Given a jpeg image taken from a phone camera, produce a higher-quality image of the same dimension.

Problem Formation: The image quality can be evaluated in many aspects, including (1) noise level, (2) resolution and sharpness, (3) brightness and contrast, (4) color correction, (5) occlusion etc. Instead of an ad-hoc solution to improve any one of the aspects, I assume *a general image restoration model* is desired in this problem. As the diffusion denoising probabilistic models (DDPMs) have achieved better performance in image generation compared with generative adversarial networks, **I present a method to inject the conditional low-quality image to a well-trained unconditional DDPM G_Θ. Trained on huge high-quality (HQ) data, G_Θ has the prior knowledge of HQ data distribution. The core problem to solve is: how to apply the conditional constraint provided by y .**

I first formulate the image degradation function by a non-linear model F_Φ with learnable parameters Φ:

$$\mathbf{y} = F_\Phi(\mathbf{x}) \tag{1}$$

where $\mathbf{y}$ is the degraded image, $\mathbf{x}$ is the unknown original image, and $\mathbf{y}, \mathbf{x} \in \mathbb{R}^{3\times h\times w}$. Although many of the image degradation can be regarded as linear (i.e., $\mathbf{y} = H\mathbf{x}+M$), there are many non-linear operations such as low-light enhancement and contrast adjustment.

$\mathbf{y}$	H	M
noisy	identical matrix I	noise matrix
blurry	Gaussian blur kernel	$\mathbf{0}$
low resolution	downsample operator	$\mathbf{0}$
occlusion	binary mask	content (e.g., text)

Table 1: Examples of linear degradation

Apparently, the learnable model is able to inclusively delineate more types on scenarios. Note that for the k^{th} type of degradation, there should be a corresponding model F_{Φ_k} where $k = 1, \cdots, K$. During training, we need to optimize the model parameters Φ_k such that the predicted image $\tilde{\mathbf{x}}$ has higher quality and its content is consistent with the input $\mathbf{y}$.

Background: Suppose the noise variance schedule is $\{\beta_1, \beta_2, \cdots, \beta_T\}$ where T is the total number of denoising time steps. And we define $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_i^t \alpha_t$. Note that all these values are pre-defined constants.

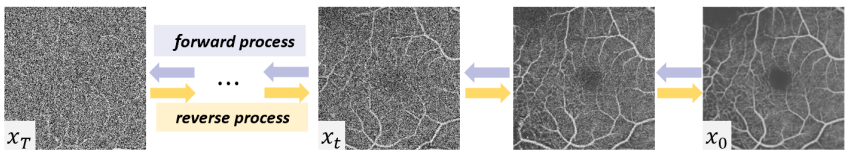

Figure 1: Diffusion probabilistic model

For the forward process (diffusion), the noisy image at time step t can be acquired by:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{2}$$

where $\mathbf{x}_0$ is the clean image and the $\epsilon \sim \mathcal{N}(0, I)$.

For the reverse process (denoise), the sampling is achieved by:

$$\mathbf{x}_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} G_{\boldsymbol{\Theta}}(\mathbf{x}_t, t)\right)}_{\mu_t} + \underbrace{\frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}_{\sigma_t}\epsilon \tag{3}$$

where $\epsilon \sim \mathcal{N}(0, I)$, and $\mathbf{x}_{t-1}$ is the one-step denoised image sampled from $\mathbf{x}_t$ and model prediction $G_\Theta(\mathbf{x}_t, t)$. Here the time step t serves as one of the input by sinusoidal time embedding. After iterative denoising, the final model prediction will be $\tilde{\mathbf{x}} = \mathbf{x}_0$. Intuitively, diffusion probabilistic model is mapping a Gaussian distribution to a complex data distribution by a Markov chain with T small steps, which greatly simplify the direct mapping. Each step $p_\Theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is equivalent to sample from a distribution $\mathcal{N}(\mu_t, \sigma_t^2)$.

In practice, there is an alternative expression for $p_\Theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ which is:

$$\mathbf{x}_{t-1} = \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_{0|t} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t}_{\mu_t} + \underbrace{\frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}_{\sigma_t}\epsilon \tag{4}$$

where $\epsilon \sim \mathcal{N}(0, I)$, and $\mathbf{x}_{0|t}$ is the prediction of $\mathbf{x}_0$ directly from $\mathbf{x}_t$:

$$\mathbf{x}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} G_{\boldsymbol{\Theta}}(\mathbf{x}_t, t)\right) \tag{5}$$

Method: To tackle the aforementioned image restoration problem, we need to impose the degraded image $\mathbf{y}$ as an extra conditional term i.e., $p_\Theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$. It has been proved that

$$\log p_\Theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) = \log\left(p_\Theta(\mathbf{x}_{t-1}|\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_t)\right) \approx \log p(\mathbf{r}) \tag{6}$$

where $\mathbf{r} \sim \mathcal{N}(\mu_t + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t), \sigma_t)$. The term $p(\mathbf{y}|\mathbf{x}_t)$ can be regarded as the probability of $\mathbf{x}_t$ be denoised to a high-quality image consistent to $\mathbf{y}$. Therefore, the shift of the mean value during sampling is modelled by the gradient of the loss function:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \mathcal{L}(\mathbf{y}, F_\Phi(\mathbf{x}_t)) \tag{7}$$

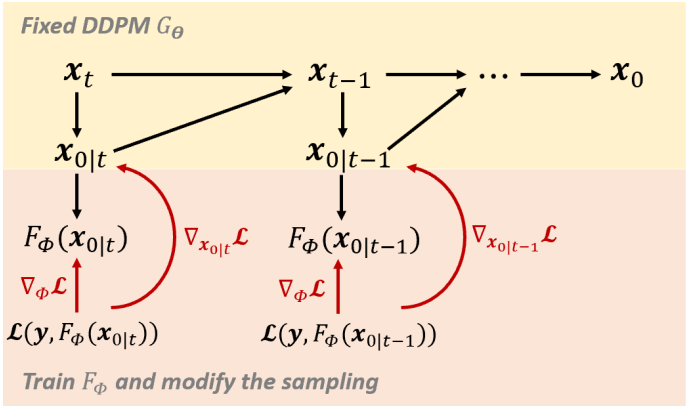

Figure 2: Proposed method

Since $\mathbf{x}_t$ inherently contains Gaussian noise, the conditional signal $\mathbf{y}$ can be instead conducted on $\mathbf{x}_{0|t}$. The loss function is shown as following:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{fidelity}(\mathbf{y}, F_\Phi(\mathbf{x}_{0|t})) + \lambda_2 \mathcal{L}^k_{perceptual}(\mathbf{x}_{0|t}) \tag{8}$$

input : low-quality image $\mathbf{y}$, randomly initialized parameters Φ for the degradation model F_Φ, well-trained diffusion model G_Θ, the variance schedule $\{\beta_1, \cdots, \beta_T\}$, learning rate η, gradient step s, loss function $\mathcal{L}_{total}$ with weight λ_1 and λ_2

output: high-quality image $\mathbf{x}_0$

Sample $\mathbf{x}_T$ from $\mathcal{N}(0, I)$

for $t = T : 1$ **do**

  $\mathbf{x}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} G_{\mathbf{\Theta}}(\mathbf{x}_t, t) \right)$ // Eq.5

  $\mu_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \mathbf{x}_{0|t} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}_t$ // Eq.4

  $\sigma_t = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$ // Eq.4

  $\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{fidelity}(\mathbf{y}, F_\Phi(\mathbf{x}_{0|t})) + \lambda_2 \mathcal{L}^k_{perceptual}(\mathbf{x}_{0|t})$ // Eq.8

  $\Phi \leftarrow \Phi - \eta \nabla_\Phi \mathcal{L}_{total}$

  $\mathbf{x}_{t-1} = \left(\mu_t + s \nabla_{\mathbf{x}_{0|t}} \mathcal{L}_{total} \right) + \sigma_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$

return $\mathbf{x}_0$

Fidelity loss. The first term $\mathcal{L}_{fidelity}$ measures the distance between two images, it is used to control the content of generated image $\mathbf{x}_{0|t}$ to be consistent with $\mathbf{y}$. Since the image $\mathbf{y}$ is degraded, the pixel-wise constraint cannot be directly applied on $\mathbf{x}_{0|t}$. We first map $\mathbf{x}_{0|t}$ to the degradation manifold with the linear projection $H\mathbf{x}_{0|t} + M$. In practice, this fidelity loss can be realized by MSE:

$$\mathcal{L}_{fidelity}(\mathbf{y}, F_\Phi(\mathbf{x}_{0|t})) = \|\mathbf{y} - F_\Phi(\mathbf{x}_{0|t})\|_2^2 \tag{9}$$

Perceptual loss. The second loss function $\mathcal{L}_{perceptual}$ is an optional term that ensure the predicted image $\mathbf{x}_{0|t}$ is photo-realistic. For example, this can be achieved by implementing a pre-trained network like VGG-16. Moreover, this loss function can be specifically designed for different types of degradation, so I used the superscript k.

Training data preparation: Suppose we are using the high-quality image datasets such as the **ImageNet**, **LSUN** and **CelebA**, then we degrade the images with known linear functions.

- Super-resolution: apply a block averaging filter to downscale the image on each axis d times. Usually $d = 4$ or $d = 8$.
- Deblurring: blur the image with a $n \times n$ Gaussian kernel.
- Colorization: take the average of the red, green and blue channels of the orifinal image
- Inpainting: Randomly generate a binary mask with 25% of the pixels are set to zero. Then conduct the Hadamard product with the original image.

For other non-linear scenarios, the images from the low-light dataset and **NTIRE** dataset are naturally over-exposed or under-exposed. No additional operations needed.