Diffusion Application on low-level vision

What is low-level？

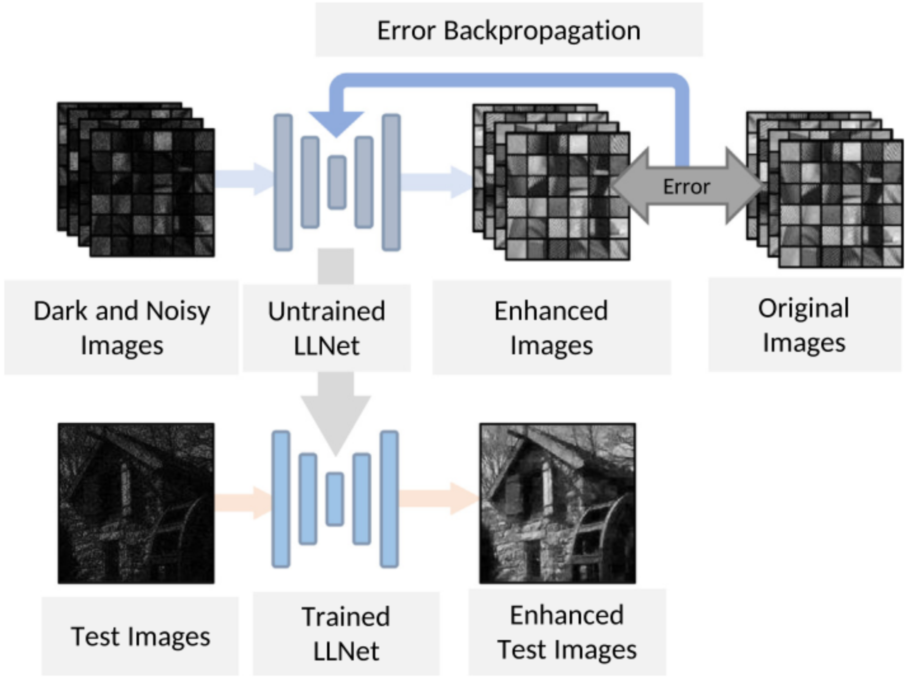
Low-level task aims to convert degraded images to enhanced images, such as Super-Resolution, denoise,deblur,dehaze,low-light enhancement, deartifacts…….

Supervised Method:

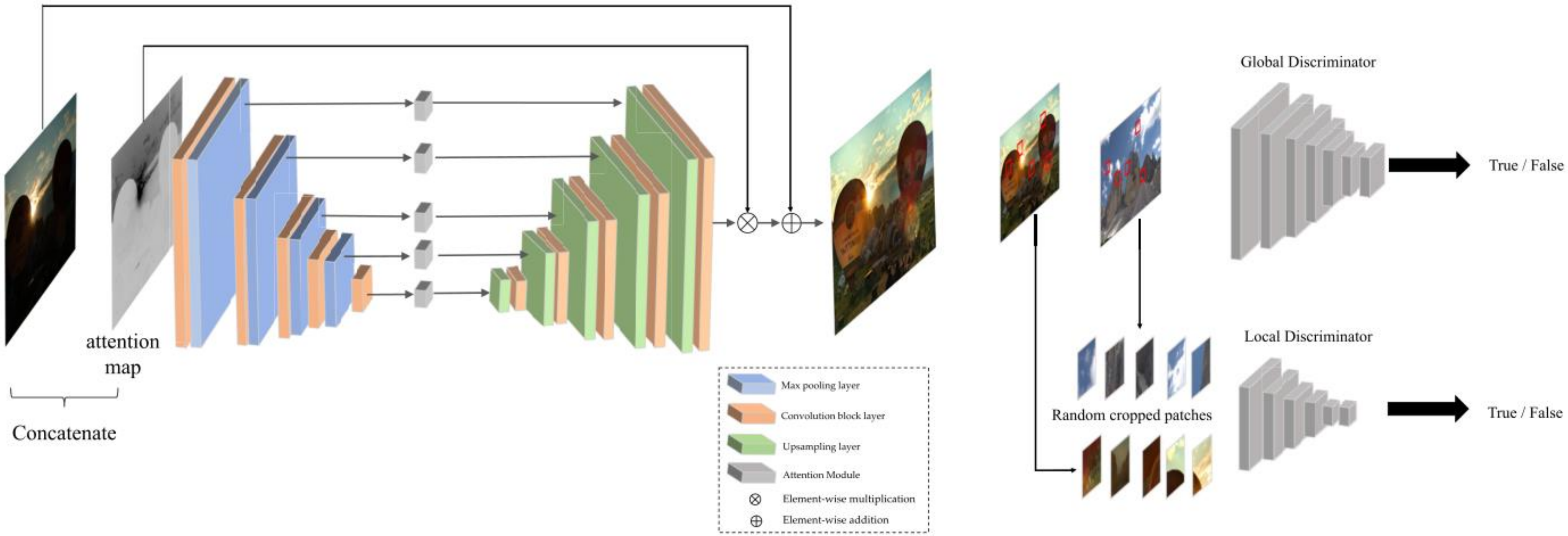
Deep neural network

Paired data

Encoder-Decoder



GAN Method



attention map
Concatenate
Max pooling layer
Convolution block layer
Upsampling layer
Attention Module
Element-wise multiplication
Element-wise addition
Global Discriminator
True / False
Local Discriminator
True / False
Random cropped patches

Palette: Image-to-Image Diffusion Models

Chitwan Saharia, William Chan, Huiwen Chang, Chris A Lee, Jonathan Ho, Tim Salimans
David J Fleet, Mohammad Norouzi
Google Research, Brain Team
Canada
{sahariac,williamchan,davidfleet,mnorouzi}@google.com

Abstract

Palette develops a unified framework for image-to-image translation based on conditional diffusion models and evaluates this framework on four challenging image-to-image translation tasks, namely colorization, inpainting, uncropping, and JPEG restoration.

This implementation of image-to-image diffusion models out performs strong GAN and regression baselines on all tasks, without task-specific hyper-parameter tuning, architecture customization, or any auxiliary loss or sophisticated new techniques needed.

Method

x： degraded image

y： enhanced image

Origin Diffusion

The forward diffusion process is a Markovian process that iteratively adds Gaussia noise over T iterations:

$$
\begin{aligned}
q(\boldsymbol{y}_{t+1}|\boldsymbol{y}_t) &= \mathcal{N}(\boldsymbol{y}_{t-1}; \sqrt{\alpha_t}\boldsymbol{y}_{t-1}, (1-\alpha_t)I) && (2)\\
q(\boldsymbol{y}_{1:T}|\boldsymbol{y}_0) &= \prod_{t=1}^{T} q(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}) && (3)
\end{aligned}
$$

Note, we can also marginalize the forward process at each step:

$$
q(\boldsymbol{y}_t|\boldsymbol{y}_0) = \mathcal{N}(\boldsymbol{y}_t; \sqrt{\gamma_t}\boldsymbol{y}_0, (1-\gamma_t)I)\ , \tag{4}
$$

where $\gamma_t = \prod_{t'}^{t} \alpha'_t$.

Use Bayer's rule:

$$
q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_0, \boldsymbol{y}_t) = \mathcal{N}(\boldsymbol{y}_{t-1} \mid \boldsymbol{\mu}, \sigma^2\boldsymbol{I}) \tag{5}
$$

where $\boldsymbol{\mu} = \frac{\sqrt{\gamma_{t-1}}\,(1-\alpha_t)}{1-\gamma_t}\boxed{\boldsymbol{y}_0} + \frac{\sqrt{\alpha_t}\,(1-\gamma_{t-1})}{1-\gamma_t}\boldsymbol{y}_t$ and $\sigma^2 = \frac{(1-\gamma_{t-1})(1-\alpha_t)}{1-\gamma_t}$.

Palette Training:

Palette learns a reverse process which inverts the forward process. Given a noisy image $\widetilde{\boldsymbol{y}}$,

$$\widetilde{\boldsymbol{y}} = \sqrt{\gamma}\, \boldsymbol{y}_0 + \sqrt{1-\gamma}\, \boldsymbol{\epsilon}\,, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})\,, \tag{6}$$

the goal is to recover the target image y_0

Learning entails prediction of the noise vector ε by optimizing the objective

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\mathbb{E}_{\boldsymbol{\epsilon},\gamma}\bigg\| f_\theta(\boldsymbol{x}, \underbrace{\sqrt{\gamma}\, \boldsymbol{y}_0 + \sqrt{1-\gamma}\, \boldsymbol{\epsilon}}_{\widetilde{\boldsymbol{y}}}, \gamma) - \boldsymbol{\epsilon} \bigg\|_p^p \,.$$

Palette Sample:

given y_t , we approximate y_0 by rearranging terms in equation 6

$$\hat{\boldsymbol{y}}_0 = \frac{1}{\sqrt{\gamma_t}}\left(\boldsymbol{y}_t - \sqrt{1-\gamma_t}\, f_\theta(\boldsymbol{x}, \boldsymbol{y}_t, \gamma_t)\right) .$$

With this parameterization, each iteration of the reverse process can be computed as

$$\boldsymbol{y}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta(\boldsymbol{x}, \boldsymbol{y}_t, \gamma_t)\right) + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t\,,$$

Palette

Algorithm 1 Training a denoising model f_θ

repeat
  $(\boldsymbol{x}, \boldsymbol{y}_0) \sim p(\boldsymbol{x}, \boldsymbol{y})$
  $\gamma \sim p(\gamma)$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take a gradient descent step on
  $$\nabla_\theta \left\| f_\theta(\boldsymbol{x}, \sqrt{\gamma}\boldsymbol{y}_0 + \sqrt{1-\gamma}\boldsymbol{\epsilon}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p$$
until converged

Algorithm 2 Inference in T iterative refinement steps

$\boldsymbol{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
  $\boldsymbol{y}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta(\boldsymbol{x}, \boldsymbol{y}_t, \gamma_t)\right) + \sqrt{1-\alpha_t}\boldsymbol{z}$
end for
return $\boldsymbol{y}_0$

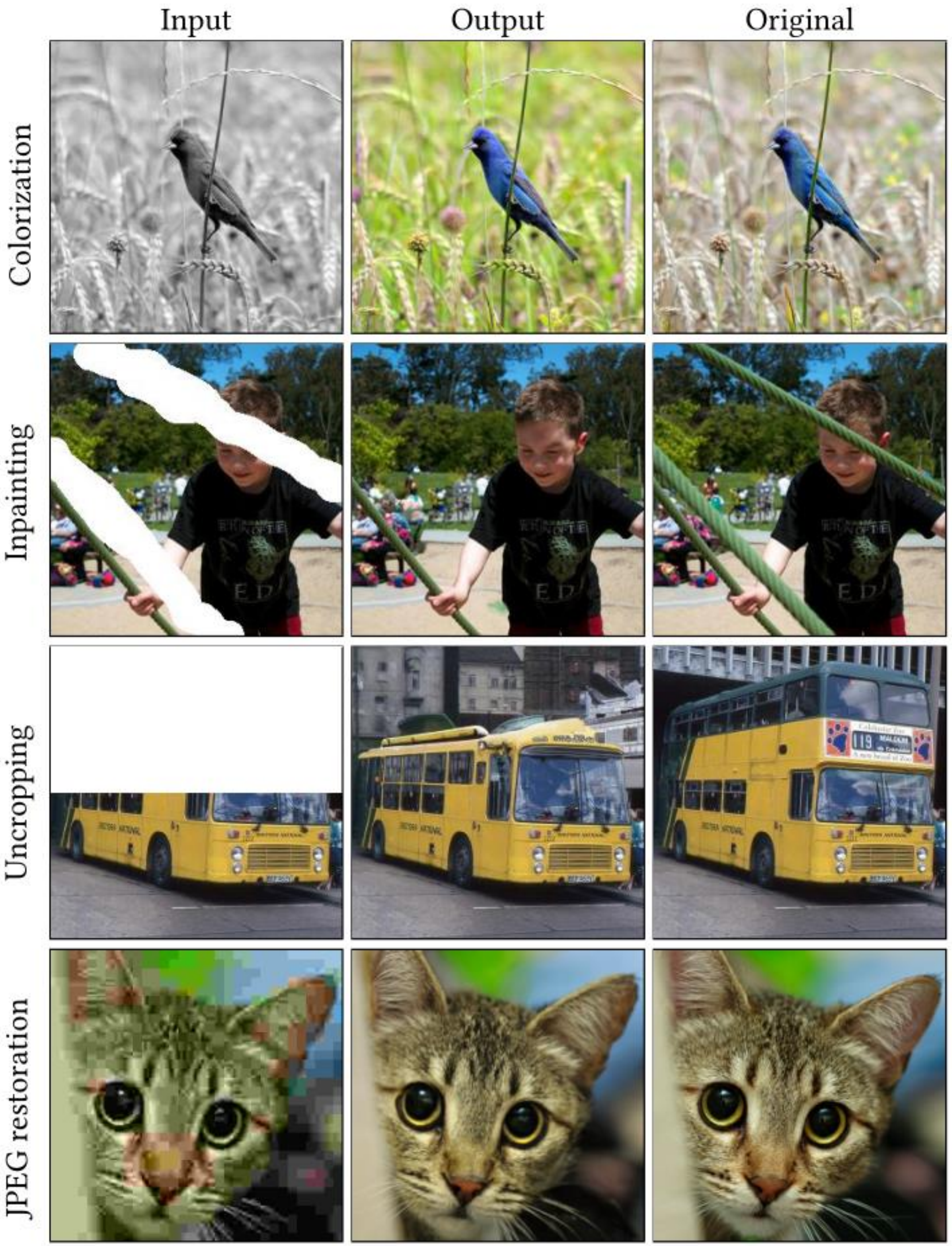
DDPM

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
  $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$



Model	FID-5K ↓	IS ↑	CA ↑	PD ↓	Fool rate ↑
Prior Work					
pix2pix †	24.41	-	-	-	-
PixColor ‡	24.32	-	-	-	29.90%
Coltran ††	19.37	-	-	-	36.55%
This paper					
Regression	17.89	169.8	68.2%	60.0	39.45%
Palette	**15.78**	**200.8**	**72.5%**	**46.2**	**47.80%**
Original images	14.68	229.6	75.6%	0.0	-

Table 1: Colorization quantitative scores and fool rates on ImageNet val set indicate that Palette outputs are bridging

Model	ImageNet				Places2	
	FID ↓	IS ↑	CA ↑	PD ↓	FID ↓	PD ↓
20-30% free form						
DeepFillv2 [Yu et al. 2019]	9.4	174.6	68.8%	64.7	13.5	63.0
HiFill [Yi et al. 2020]	12.4	157.0	65.7%	86.2	15.7	92.8
Co-ModGAN [Zhao et al. 2021]	-	-	-	-	12.4	51.6
Palette (Ours)	**5.2**	**205.5**	**72.3%**	**27.6**	**11.7**	**35.0**
128×128 center						
DeepFillv2 [Yu et al. 2019]	18.0	135.3	64.3%	117.2	15.3	96.3
HiFill [Yi et al. 2020]	20.1	126.8	62.3%	129.7	16.9	115.4
Co-ModGAN [Zhao et al. 2021]	-	-	-	-	13.7	86.2
Palette (Ours)	**6.6**	**173.9**	**69.3%**	**59.5**	**11.9**	**57.3**
Original images	5.1	231.6	74.6%	0.0	11.4	0.0

Table 2: Quantitative evaluation for free-form and center in-painting on ImageNet and Places2 validation images.

Denoising Diffusion Restoration Models(DDRM)

Background:

- Previous efficient solutions often require problem-specific supervised training to model the posterior.
- DDRM is an efficient, not problem-specific ,unsupervised posterior sampling method.

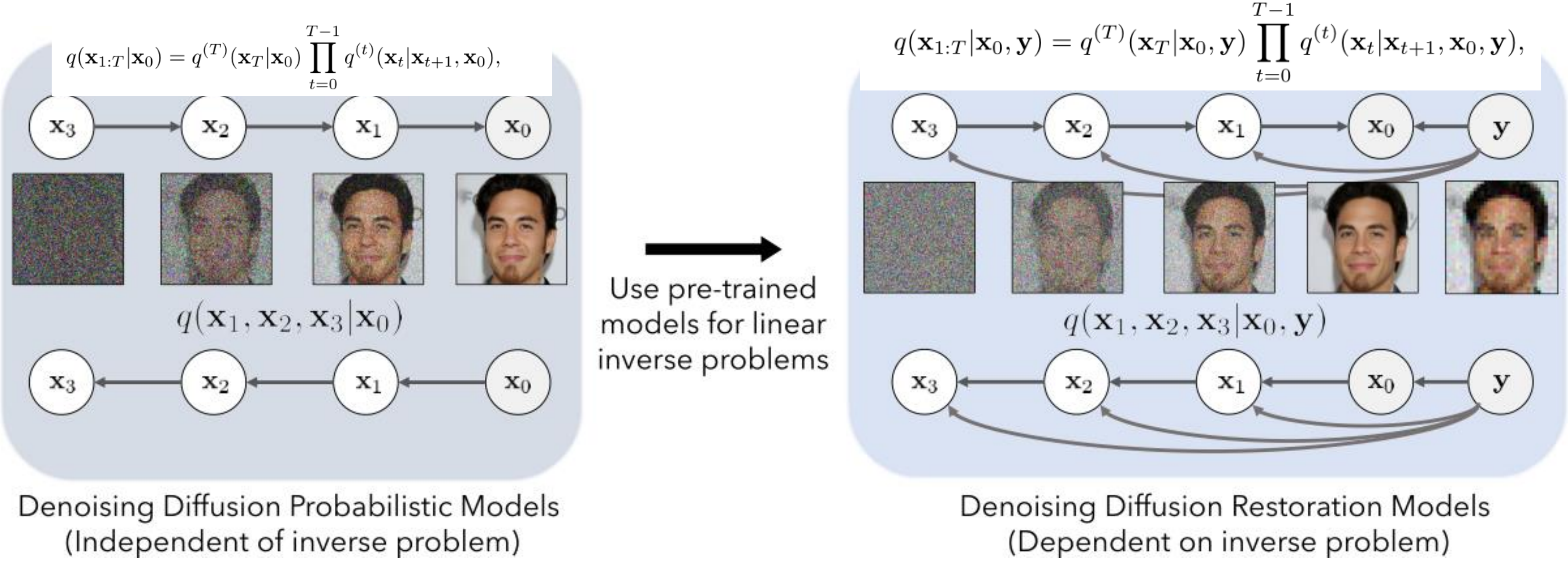
Problem Define:

Many tasks in image restoration can be cast as linear inverse problems.

$y = Hx + z,$

$where\ H\ is\ a\ known\ linear\ degradation\ matrix, z \sim N(0, \sigma^2 I)$

Main idea of DDRM:



DDRM is to leverage the singular value decomposition of H and transform both x and the possibly noisy y to a shared spectral space.

$$H = U\Sigma V^\top \qquad \bar{\mathbf{x}}_t = V^\top \mathbf{x}_t \qquad \bar{\mathbf{y}} = \Sigma^\dagger U^\top \mathbf{y}$$

Variational distribution

$$p_\theta^{(T)}(\bar{\mathbf{x}}_T^{(i)}|\mathbf{y}) = \begin{cases} \mathcal{N}(\bar{\mathbf{y}}^{(i)}, \sigma_T^2 - \frac{\sigma_{\mathbf{y}}^2}{s_i^2}) & \text{if } s_i > 0 \\ \mathcal{N}(0, \sigma_T^2) & \text{if } s_i = 0 \end{cases} \quad (7)$$

$$p_\theta^{(t)}(\bar{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t+1}, \mathbf{y}) = \quad (8)$$

$$\begin{cases} \mathcal{N}(\bar{\mathbf{x}}_{\theta,t}^{(i)} + \sqrt{1-\eta^2}\sigma_t \frac{\bar{\mathbf{x}}_{t+1}^{(i)} - \bar{\mathbf{x}}_{\theta,t}^{(i)}}{\sigma_{t+1}}, \eta^2\sigma_t^2) & \text{if } s_i = 0 \\ \mathcal{N}(\bar{\mathbf{x}}_{\theta,t}^{(i)} + \sqrt{1-\eta^2}\sigma_t \frac{\bar{\mathbf{y}}^{(i)} - \bar{\mathbf{x}}_{\theta,t}^{(i)}}{\sigma_{\mathbf{y}}/s_i}, \eta^2\sigma_t^2) & \text{if } \sigma_t < \frac{\sigma_{\mathbf{y}}}{s_i} \\ \mathcal{N}((1-\eta_b)\bar{\mathbf{x}}_{\theta,t}^{(i)} + \eta_b\bar{\mathbf{y}}^{(i)}, \sigma_t^2 - \frac{\sigma_{\mathbf{y}}^2}{s_i^2}\eta_b^2) & \text{if } \sigma_t \geq \frac{\sigma_{\mathbf{y}}}{s_i}. \end{cases}$$

Reverse diffusion process

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

$\bar{\mathbf{x}}_{\theta,t}^{(i)}$ represent this prediction made by a model $f(x_{t+1}, t+1): R^{n*n} \times R \rightarrow R^{n*n}$

JPEG Artifact Correction using Denoising Diffusion Restoration Models

---

Bahjat Kawar[*]
Technion, Haifa, Israel
bahjat.kawar@cs.technion.ac.il

Jiaming Song[*]
NVIDIA
jiamings@nvidia.com

Stefano Ermon
Stanford University, CA
ermon@cs.stanford.edu

Michael Elad
Technion, Haifa, Israel
elad@cs.technion.ac.il

Non-linear DDRM

For the case of no noise in the observation $\mathbf{y}$, the $\boxed{\text{general DDRM process}}$ to sample from $p_\theta^{(t)}(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y})$ for linear inverse problems simplifies to be

$$\mathbf{x}'_t = f_\theta^{(t+1)}(\mathbf{x}_{t+1}) - \boldsymbol{H}^\dagger \boldsymbol{H} f_\theta^{(t+1)}(\mathbf{x}_{t+1}) + \boldsymbol{H}^\dagger \mathbf{y},$$

$$\mathbf{x}_t = \sqrt{\alpha_t}\left(\eta_b \mathbf{x}'_t + (1-\eta_b) f_\theta^{(t+1)}(\mathbf{x}_{t+1})\right) + \sqrt{1-\alpha_t}\left(\eta \epsilon_t + (1-\eta)\epsilon_\theta^{(t+1)}(\mathbf{x}_{t+1})\right)$$

---

if we treat H as the JPEG encoding operator, H^+ as the JPEG decode operator

$f_\theta^{(t+1)}(X_{t+1}) = X_0 = \frac{1}{\sqrt{\bar\alpha_t}}(X_t - \sqrt{1-\bar\alpha_t}\,\epsilon_t)$

model(X_t, t)

predicted noise

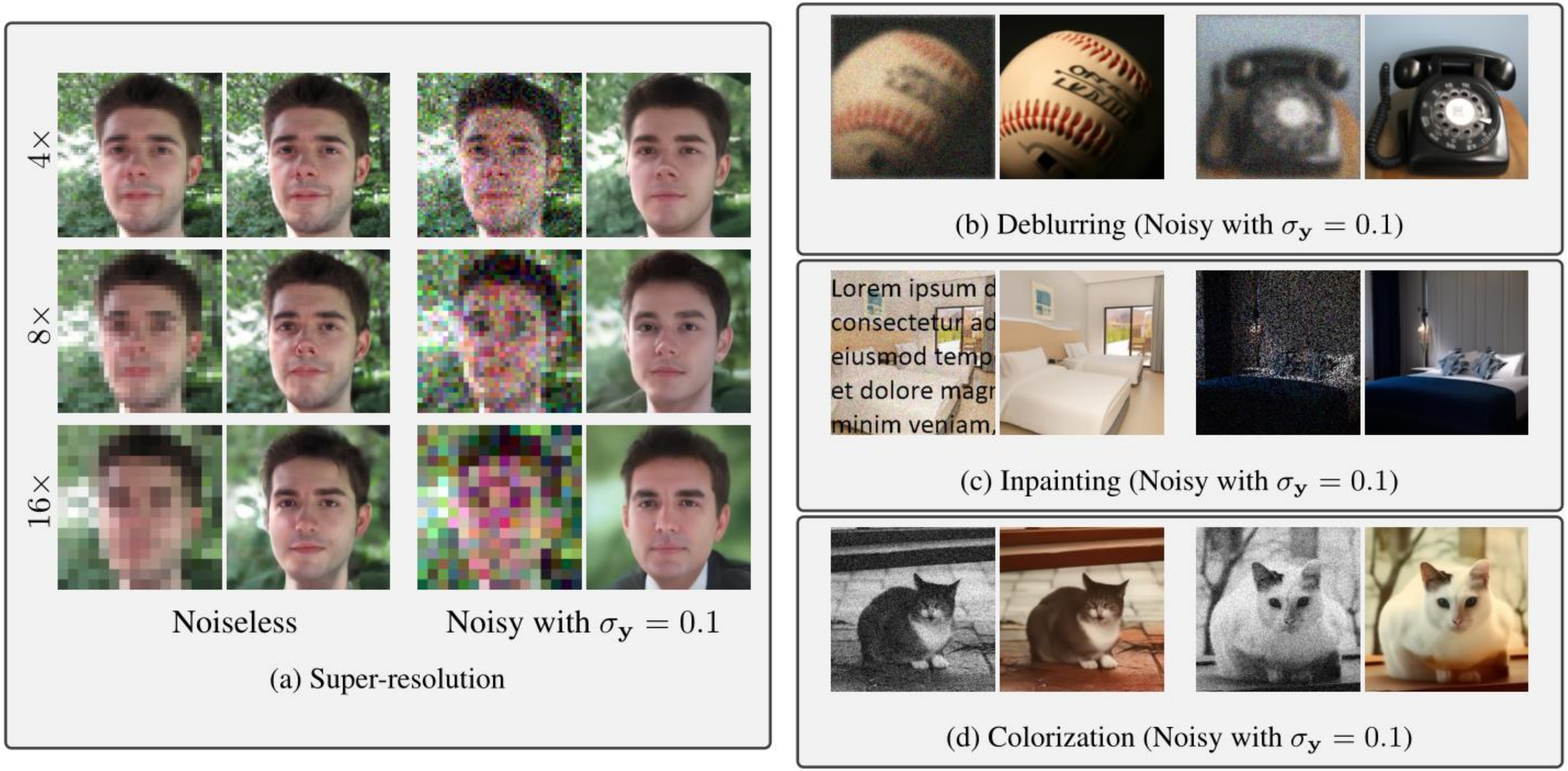
With this insight, we can simply perform JPEG restoration with DDRM with the update rule

$$\mathbf{x}'_t = \underline{f_\theta^{(t+1)}(\mathbf{x}_{t+1})} - \text{Decode}\left(\text{Encode}\left(f_\theta^{(t+1)}(\mathbf{x}_{t+1})\right)\right) + \text{Decode}(\mathbf{y}),$$

$$\mathbf{x}_t = \sqrt{\alpha_t}\left(\eta_b \mathbf{x}'_t + (1-\eta_b) f_\theta^{(t+1)}(\mathbf{x}_{t+1})\right) + \sqrt{1-\alpha_t}\left(\eta \epsilon_t + (1-\eta)\epsilon_\theta^{(t+1)}(\mathbf{x}_{t+1})\right),$$

which can be used in realistic settings as the quantization matrices are stored within the JPEG files.

$\mathcal{N}(0, I)$



(a) Super-resolution

(b) Deblurring (Noisy with $\sigma_{\mathbf{y}} = 0.1$)

(c) Inpainting (Noisy with $\sigma_{\mathbf{y}} = 0.1$)

(d) Colorization (Noisy with $\sigma_{\mathbf{y}} = 0.1$)