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
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 outperforms 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 \hspace{1cm} \( y \): enhanced image

Origin Diffusion

The forward diffusion process is a Markovian process that iteratively adds Gaussian noise over \( T \) iterations:

\[
q(y_{t+1}|y_t) = N(y_{t-1}; \sqrt{\alpha_t} y_{t-1}, (1 - \alpha_t)I) \tag{2}
\]

\[
q(y_{1:T}|y_0) = \prod_{t=1}^{T} q(y_t|y_{t-1}) \tag{3}
\]

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

\[
q(y_t|y_0) = N(y_t; \sqrt{\gamma_t} y_0, (1 - \gamma_t)I) , \tag{4}
\]

where \( \gamma_t = \prod_{t} \alpha_t \).

Use Bayer’s rule:

\[
q(y_{t-1} \mid y_0, y_t) = N(y_{t-1} \mid \mu, \sigma^2 I) \tag{5}
\]

where \( \mu = \frac{\sqrt{\gamma_{t-1}}}{1 - \gamma_t} y_0 + \frac{\sqrt{\alpha_t}}{1 - \gamma_t} (1 - \gamma_{t-1}) y_t \) and \( \sigma^2 = \frac{(1 - y_{t-1})(1 - \alpha_t)}{1 - \gamma_t} \).
Palette Training:

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

$$\tilde{y} = \sqrt{\gamma} y_0 + \sqrt{1 - \gamma} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$  \hspace{1cm} (6)

the goal is to recover the target image $y_0$

Learning entails prediction of the noise vector $\epsilon$ by optimizing the objective

$$\mathbb{E}_{(x,y)} \mathbb{E}_{\epsilon, y} \left\| f_{\theta}(x, \sqrt{\gamma} y_0 + \sqrt{1 - \gamma} \epsilon, \gamma) - \epsilon \right\|^p_p.$$  

Palette Sample:

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

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

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

$$y_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}} \left( y_t - \frac{1 - \alpha_t}{\sqrt{1 - \gamma_t}} f_{\theta}(x, y_t, \gamma_t) \right) + \sqrt{1 - \alpha_t} \epsilon_t.$$
Palette

**Algorithm 1** Training a denoising model $f_\theta$

1: repeat
2: $\mathbf{x}, y_0 \sim p(x, y)$
3: $y \sim p(y)$
4: $\epsilon \sim \mathcal{N}(0, 1)$
5: Take a gradient descent step on
   $$\nabla_{\theta} \left\| f_\theta (x, \sqrt{y} y_0 + \sqrt{1 - y} \epsilon, y) - \epsilon \right\|_p^p$$
6: until converged

**Algorithm 2** Inference in $T$ iterative refinement steps

1: $y_T \sim \mathcal{N}(0, 1)$
2: for $t = T, \ldots, 1$ do
3: $z \sim \mathcal{N}(0, 1)$ if $t > 1$, else $z = 0$
4: $y_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( y_t - \frac{1 - \alpha_t}{\sqrt{1 - \gamma_t}} f_\theta (x, y_t, y_t) \right) + \sqrt{1 - \alpha_t} z$
5: end for
6: return $y_0$

DDPM

**Algorithm 1** Training

1: repeat
2: $x_0 \sim q(x_0)$
3: $t \sim \text{Uniform}\{1, \ldots, T\}$
4: $\epsilon \sim \mathcal{N}(0, I)$
5: Take gradient descent step on
   $$\nabla_{\theta} \left\| \epsilon - \epsilon_\theta (\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon, t) \right\|_Q^2$$
6: until converged

**Algorithm 2** Sampling

1: $x_T \sim \mathcal{N}(0, I)$
2: for $t = T, \ldots, 1$ do
3: $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
4: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \gamma_t}} \epsilon_\theta (x_t, t) \right) + \sigma_t z$
5: end for
6: return $x_0$
Colorization

Prior Work
- pix2pix $\uparrow$ 24.41
- PixColor $\uparrow$ 24.32
- CoTran $\uparrow\uparrow$ 19.37

This paper
- Regression 17.89 169.8 68.2% 60.0 39.45%
- Palette 15.78 200.8 72.5% 46.2 47.80%

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

Inpainting

Uncropping

JPEG restoration

Table 2: Quantitative evaluation for free-form and center inpainting 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:

Denoising Diffusion Probabilistic Models (Independent of inverse problem)

Denoising Diffusion Restoration Models (Dependent on inverse problem)

Use pre-trained models for linear inverse problems

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

$$H = U \Sigma V^\top$$

$$\tilde{x}_t = V^\top x_t$$

$$\tilde{y} = \Sigma^\dagger U^\top y$$
Variational distribution

\[ p^{(T)}_\theta(x_T | y) = \begin{cases} 
\mathcal{N}(\bar{y}, \sigma_T^2 - \frac{\sigma_y^2}{s_{t+1}^2}) & \text{if } s_i > 0 \\
\mathcal{N}(0, \sigma_T^2) & \text{if } s_i = 0
\end{cases} \]  

(7)

\[ p^{(t)}_\theta(x_t^{(i)} | x_{t+1}, y) = \begin{cases} 
\mathcal{N}(\bar{x}^{(i)}_{\theta,t} + \sqrt{1 - \eta^2} \sigma_t \frac{x^{(i)}_{t+1} - \bar{x}^{(i)}_{t+1}}{\sigma_{t+1}} , \eta^2 \sigma_t^2) & \text{if } s_i = 0 \\
\mathcal{N}(\bar{x}^{(i)}_{\theta,t} + \sqrt{1 - \eta^2} \sigma_t \frac{\bar{x}^{(i)}_{\theta,t} - \bar{x}^{(i)}_{\theta,t}}{\sigma_s/s_i} , \eta^2 \sigma_t^2) & \text{if } \sigma_t < \frac{\sigma_y}{s_i} \\
\mathcal{N}((1 - \eta_b)\bar{x}^{(i)}_{\theta,t} + \eta_b \bar{y}^{(i)} , \sigma_t^2 - \frac{\sigma_y^2}{s_i^2} \eta_b^2) & \text{if } \sigma_t \geq \frac{\sigma_y}{s_i} .
\end{cases} \]  

(8)

\[ \bar{x}^{(i)}_{\theta,t} \] represent this prediction made by a model \( f(x_{t+1}, t + 1): \mathbb{R}^{n \times n} \times \mathbb{R} \to \mathbb{R}^{n \times n} \)
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 $y$, the general DDRM process to sample from $p_{\theta}^{(t)}(x_t|x_{t+1}, y)$ for linear inverse problems simplifies to be

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

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

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

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

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

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

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

(a) Super-resolution

(b) Deblurring (Noisy with $\sigma_y = 0.1$)

(c) Inpainting (Noisy with $\sigma_y = 0.1$)

(d) Colorization (Noisy with $\sigma_y = 0.1$)