CL2: Text to Image Generation using AI

Demostrator: Yisong Shen
student: Haoxiang Huang
CID: 02470313

partA: Proof of DDPM

1. Concept of DDPM

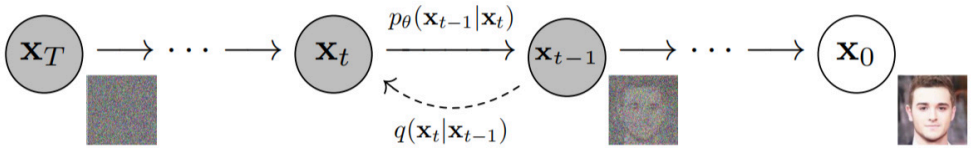


Figure 2: The directed graphical model considered in this work.

DDPM is a Parameterized Markov chains training using Variational inference to learn the transitions of this chain to reverse a diffusion process.

[learn $p_\theta(x_{t-1}|x_t)$ to denoise and generate image]

2. Diffusion Process.

2.1. One step diffusion

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$$

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \beta_t \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, I)$$

N.B. use β_t to control the speed of noise adding.

2.2. multiple steps diffusion process

We have: $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$ [Markov chain]

We use: $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

So,

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon$$

$$x_1 = \sqrt{1-\beta_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon$$
$$x_2 = \sqrt{1-\beta_2}\, x_1 + \sqrt{\beta_2}\, \varepsilon \quad \text{substitute}$$

$$x_2 = \sqrt{1-\beta_1}\sqrt{1-\beta_2}\, x_0 + \sqrt{1-\beta_2}\sqrt{\beta_1}\varepsilon + \sqrt{\beta_2}\varepsilon$$
$$= \sqrt{1-\beta_1}\sqrt{1-\beta_2}\, x_0 + \sqrt{1-(1-\beta_2)(1-\beta_1)}\,\varepsilon$$
$$\vdots$$
$$x_t = \underbrace{\sqrt{1-\beta_1}\cdots\sqrt{1-\beta_t}\, x_0}_{\sqrt{\bar{\alpha}_t}} + \underbrace{\sqrt{1-(1-\beta_1)\cdots(1-\beta_t)}}_{\sqrt{1-\bar{\alpha}_{t-1}}}\,\varepsilon.$$

$$\therefore \quad x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$$

$$q(x_t|x_0) = \mathcal{N}\left(x_t;\ \sqrt{\bar{\alpha}_t}\, x_0,\ \sqrt{1-\bar{\alpha}_t}\, I\right)$$

3. Reverse Process

Denoise

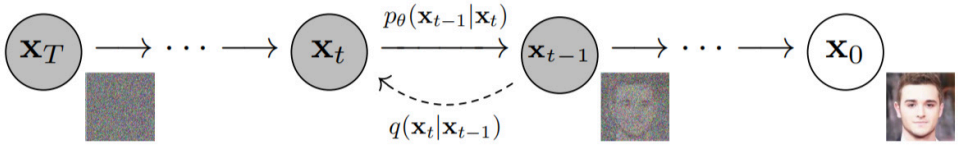


Figure 2: The directed graphical model considered in this work.

① We assume reverse process is a **Markov chain** and transitions of distribution is a **Gaussian**

⇓

② parameters estimation problem

To learn $p_\theta(x_{t-1}|x_t)$ using neural Network.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1};\ \mu_\theta(x_t, t),\ \Sigma_\theta(x_t, t)\right)$$

Note: In DDPM paper, $\Sigma_\theta(x_t, t)$ is fixed for simplity.

③ We use **Variational inference** to approximate

$p_\theta(x_{t-1}|x_t)$ to known posterior of diffusion process.

$q(x_{t-1}|x_t, x_0)$ [$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu(x_t, x_0), \tilde{\beta}_t I)$]

[proof below]

4. posterior of diffusion process.

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_{t-1}, x_t, x_0)}{q(x_t, x_0)}$$

$$= q(x_t|x_{t-1}, x_0) \cdot \frac{q(x_{t-1}|x_0)\, q(x_0)}{q(x_t|x_0)\, q(x_0)}$$

$$= q(x_t|x_{t-1}, x_0) \cdot \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$$= q(x_t|x_{t-1}) \cdot \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$$ [Markov chain]

Due to $q(x_t|x_{t-1})$, $q(x_{t-1}|x_0)$, $q(x_t|x_0)$ all are Gaussian

$q(x_t|x_{t-1}) \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$ is also Gaussian.

$$\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{\alpha}_t}\right)\right)$$

$$= \exp\left(-\frac{1}{2}\left(\underbrace{\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)}_{a} x_{t-1}^2 - \underbrace{\left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)}_{b} x_{t-1} + C\right)\right)$$

We know that $ax^2 + bx + c = 0$.

$$\Rightarrow a\left(x - \frac{b}{2a}\right)^2 + \left(\frac{4ac - b^2}{4a}\right) = 0.$$

p.d.f of Gaussian is $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$$\Rightarrow \mu = \frac{b}{2a} \qquad \sigma^2 = \frac{1}{a}$$

$$\Rightarrow \sigma^2 = 1/\left(\frac{\alpha}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t. \quad C.$$

$$\tilde{\mu}_t(x_t, x_0) = \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right) \Big/ \left(\frac{\alpha}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)$$

$$= \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right) \cdot \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0.$$

[multiple step forward]

Also, $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_t\right)$

$$\Rightarrow \mu_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right)$$

$$\Rightarrow q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\right)$$

5. Variational inference

$$KL(q(x_{1:T}|x_0) \,||\, p_\theta(x_{1:T}|x_0))$$

$$= E_{x_{1:T}\sim q}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})/p_\theta(x_0)}\right]$$

$$= E_q\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] + \log p_\theta(x_0)$$

$-ELBO$

$$\Rightarrow KL = -ELBO + \log P(x) \geq 0.$$

$$\therefore \log p_\theta(x_0) \geq ELBO$$

$$-\log p_\theta(x_0) \leq -ELBO$$

$$-E_{q(x_0)} \log p_\theta(x_0) \leq E_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

cross entropy loss. LVLB.

MIN loss $\Rightarrow$ MIN $-ELBO$ $\Rightarrow$ MIN LVLB

$$L_{VLB} = E_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

$$= E_q\left[\log \frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}\right]$$

Markov

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0)$$

$$= \frac{q(x_t, x_{t-1}, x_0)}{q(x_{t-1}, x_0)}$$

$$= \frac{q(x_{t-1}|x_t, x_0)\, q(x_t, x_0)\, q(x_0)}{q(x_{t-1})\, q(x_0)}$$

$$= \frac{q(x_{t-1}|x_t, x_0)\, q(x_t, x_0)}{q(x_{t-1})}$$

$$= E_q\left[-\log P_\theta(x_T) + \sum_{t=1}^{T} \log \frac{q(x_t|x_{t-1})}{P_\theta(x_{t-1}|x_t)}\right]$$

$$= E_q\left[-\log P_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_t|x_{t-1})}{P_\theta(x_{t-1}|x_t)} + \log \frac{q(x_1|x_0)}{P_\theta(x_0|x_1)}\right]$$

$$= E_q\left[-\log P_\theta(x_T) + \sum_{t=2}^{T} \log \left(\frac{q(x_{t-1}|x_t,x_0)}{P_\theta(x_{t-1}|x_t)} \cdot \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) + \log \frac{q(x_1|x_0)}{P_\theta(x_0|x_1)}\right]$$

$$= E_q\left[-\log P_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t,x_0)}{P_\theta(x_{t-1}|x_t)} + \sum_{t=2}^{T} \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log \frac{q(x_1|x_0)}{P_\theta(x_0|x_1)}\right]$$

$$= \log \frac{\prod q(x_t|x_0)}{\prod q(x_{t-1}|x_0)} = \log \frac{q(x_t|x_0)}{q(x_1|x_0)}$$

$$= E_q\left[-\log P_\theta(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t,x_0)}{P_\theta(x_{t-1}|x_t)} + \log \frac{q(x_T|x_0)}{\cancel{q(x_1|x_0)}} + \log \frac{\cancel{q(x_1|x_0)}}{P_\theta(x_0|x_1)}\right]$$

$$= E_q\left[\log \frac{q(x_T|x_0)}{P_\theta(x_T)} + \sum_{t=2}^{T} \log \frac{q(x_{t-1}|x_t,x_0)}{P_\theta(x_{t-1}|x_t)} - \log P_\theta(x_0|x_1)\right]$$

$$= \underbrace{D_{KL}(q(x_T|x_0)\,\|\,P_\theta(x_T))}_{L_T} + \underbrace{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t,x_0)\,\|\,P_\theta(x_{t-1}|x_t))}_{L_{T-1}} \underbrace{- \log P_\theta(x_0|x_1)}_{L_0}$$

L_T: all is Gaussian, is not related to optimization.

L_0: $t=1$, $D_K(\underbrace{q(x_0|x_t,x_0)}_{=1}\,\|\,P_\theta(x_0|x_1)) = \log P_\theta(x_0|x_1)$

It is specific L_{T-1}

So, we only care about L_{T-1}

In this way, MIN LVLB => MIN $D_{KL}(q(x_{t-1}|x_t,x_0)\,\|\,P_\theta(x_{t-1}|x_t))$

$\Rightarrow$ approximate $p_\theta(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

We have $KL(P,q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$

σ_p, σ_q is constant, only $\frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2}$ can contribute to KL

$$KL_t = \frac{1}{2\sigma_q^2}\left\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|^2$$

$\therefore$ MIN KL $\Rightarrow$ MIN $\tilde{\mu}_t$ and μ_θ.

Also, $\mu_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{1-\bar{\alpha}_t}\varepsilon_t\right)$

x_t is known.

We only need to train neural network to predict ε_t

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{1-\bar{\alpha}_t}\varepsilon_\theta(x_t, t)\right)$$

$$\Rightarrow KL = \frac{1}{2\sigma_q^2}\left\| \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)\right)\right\|^2$$

$$= \frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\sigma_q^2}\left\|\varepsilon_t - \varepsilon_\theta(x_t, t)\right\|^2$$

C.

$\because x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t$

$\therefore KL = C \cdot \left\|\varepsilon_t - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, t\right)\right\|^2$

In DDPM original paper, author find $C=1$ doesn't

affect the result for simplify.

$\therefore$ MIN $D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \Rightarrow$ MIN $\|\epsilon_t - \epsilon_\theta\|$

we train NN to predict ϵ_t.

Based Above, we know:

MIN loss $\Rightarrow$ MIN LVLB $\Rightarrow$ MIN $\|\epsilon_t - \epsilon_\theta\|$

$\therefore$ loss $= \| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t) \|^2$

FINally:

we get two algorithm: : training and Sampling:

Algorithm 1 Training

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

Algorithm 2 Sampling

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

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\right)$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\right)$$

Part B: Implementation of DDPM

1. set up hyper parameters

1.1 Hyperparameters

```python
# new hyperparams
TIMESTEPS = 1000
beta_0 = 0.0001
beta_T = 0.02
LEARNING_RATE = 0.001
# generate alphas,betas,alpha_bars using my unique name hhx to store
hhx_betas = torch.linspace(beta_0, beta_T, TIMESTEPS, device = DEVICE)
hhx_alphas = 1.0 - hhx_betas
hhx_alpha_bars = torch.cumprod(hhx_alphas, axis=0)
```

→ $\beta_0 \sim \beta_t \in [0.02, 10^{-4}]$ based on DDPM paper.

$$\begin{cases} \alpha = 1 - \beta_t \\ \bar{\alpha} = \prod_{i=1}^{t} \alpha_i \end{cases}$$

2. Diffusion process.

2.1 one step Diffusion process.

```python
def fixed_one_step_forward(xt_minus_1, t):
    """
    fixed one step of a forward process according to one step formula in DDP

    Args:
    xt_minus_1(torch.Tensor): The input image tensor in t-1 step (previous i
    t (int): The current time step.

    Returns:
    xt (torch.Tensor): The output image tensor in t step after one step of t

    """

    noise = torch.randn_like(xt_minus_1, device = DEVICE)
    beta_t = hhx_betas[t]
    xt= torch.sqrt(1.0 - beta_t) * xt_minus_1 + torch.sqrt(beta_t) * noise

    return xt
```

→ $\varepsilon \sim N(0, I)$

→ $x_t = \sqrt{1-\beta_t}\, x_{t-1} + \beta_t \varepsilon_t$

$= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\, \varepsilon_t$.

2.2. multiple steps. diffusion process.

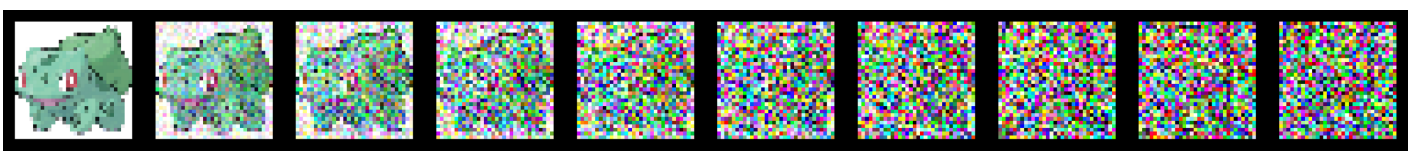
```python
def hhx_forward_diffusion(x_0, t):
    """
    Perform the forward diffusion process based on the given input clean image x_0
    This function computes x[t] based on x[0], using the diffusion process defined

    Args:
    - x_0 (torch.Tensor): The input clean image tensor. Expected to be a 4D tensor
                          (batch_size, Channels, Img_size, Img_size).
    - t (torch.Tensor): The timestep tensor.

    Returns:
    - torch.Tensor: The tensor x[t] after the diffusion process.
    - torch.Tensor: The noise tensor generated during the process.
    """
    # Generate a sample from the normal distribution
    noise = torch.randn_like(x_0)
    # must be 4 dim batch_size X Channels X Img_size X Img_size
    if not x_0.dim() == 4:
        assert(False)
    # Get the mean and variance. Choose sqrt_alphas_bar_t and sqrt_one_minus_alpha
    # values based on the timestep.
    sqrt_alphas_bar_t = torch.sqrt(torch.gather(hhx_alpha_bars, 0, t))
    sqrt_one_minus_alphas_bar = torch.sqrt(1-torch.gather(hhx_alpha_bars, 0, t))
    batch_dim = x_0.size(0)

    # Compute x[t] based on x[0]
    x_t = sqrt_alphas_bar_t.view((batch_dim,1,1,1)) * x_0 + \
                sqrt_one_minus_alphas_bar.view(batch_dim,1,1,1) * noise

    return x_t ,noise
```

$\sqrt{\bar{\alpha}_t} = \sqrt{\prod_{i=1}^{t} \alpha_i}$

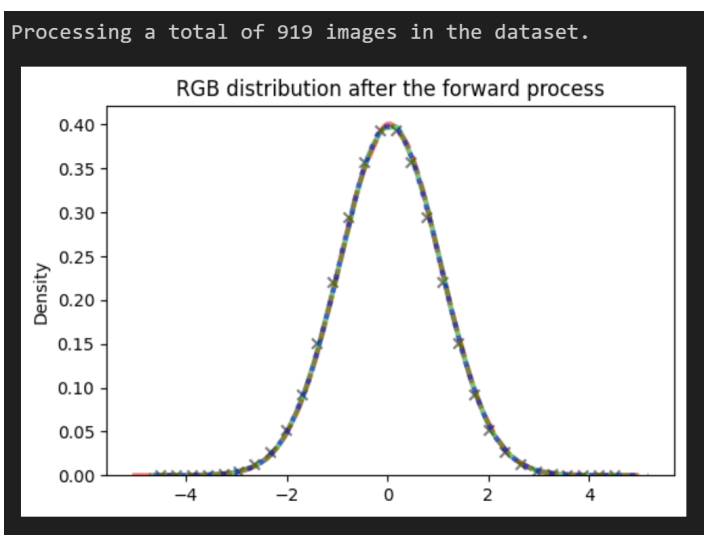
$X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$

forward visulize :

$\beta_t \uparrow \quad \bar{\alpha}_t \downarrow$



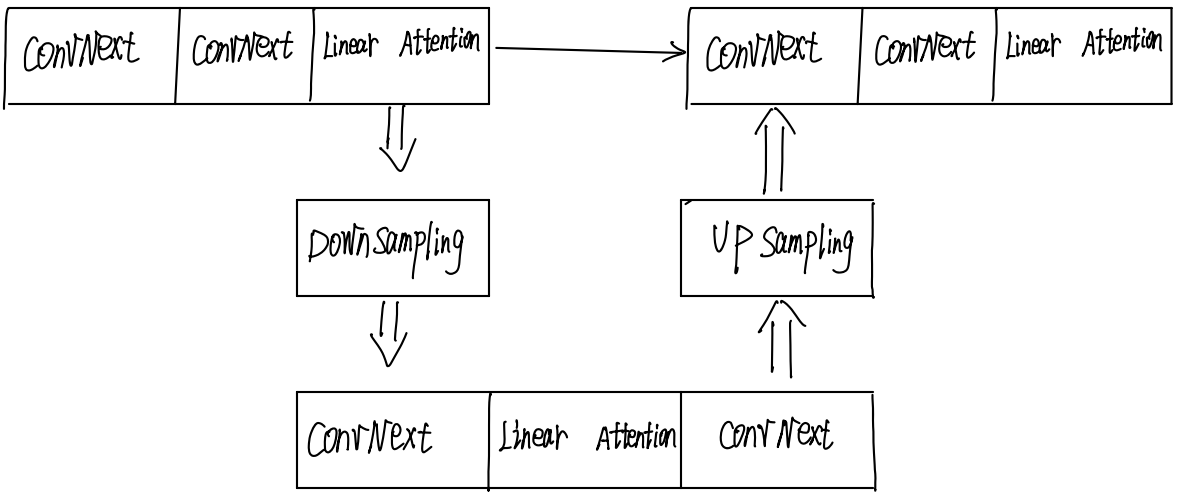
X_0 X_T



RGB statistics follows standard Gaussian distribution

3. Diffusion Backward Noise Predict model

[UNet based on ConvNext Block]



4. One step Sampling

```python
def denoise_one_step_sample(model, x, timestep):
    beta_t = hhx_betas[timestep]
    alpha_t = hhx_alphas[timestep]
    alpha_bar_t = hhx_alpha_bars[timestep]
    sqrt_one_minus_alpha_bar_t = torch.sqrt(1-alpha_bar_t)
    sqrt_recip_alphas_t = torch.sqrt(1.0 / alpha_t)

    timestep_tensor = torch.tensor([timestep], device=DEVICE)
    model_mean = sqrt_recip_alphas_t * (
        x - beta_t * model(x,  timestep_tensor) / sqrt_one_minus_alpha_bar_t)

    alpha_bar_prev = F.pad(hhx_alpha_bars[:-1], (1, 0), value=1.0)

    # calculations for posterior q(x_{t-1} | x_t, x_0)
    posterior_variance = hhx_betas * (1. - alpha_bar_prev) / (1. - hhx_alpha_bars)

    if timestep == 0:
        return model_mean
    else:
        posterior_variance_t = posterior_variance[timestep]
        noise = torch.randn_like(x)
        return model_mean + torch.sqrt(posterior_variance_t) * noise
```

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)\right) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\varepsilon$$

$$\varepsilon \sim N(0, I)$$

5. trainning

```python
def train_epoch(model, dataloader, optimizer):
    loss_list = list()
    for x in dataloader:
        x= x.to(DEVICE)
        batch_size = x.shape[0]
        optimizer.zero_grad()
        t = torch.randint(0, TIMESTEPS - 1, (batch_size,), device = DEVICE)
        x_t, noise = hhx_forward_diffusion(x, t)
        noise_pred = model(x_t, t)
        noise = noise.squeeze()
        # print("noise")
        # print(noise.shape)
        # print("noise_pred")
        # print(noise_pred.shape)
        loss = F.mse_loss(noise_pred, noise)
        loss.backward()
        loss_list.append(loss.item())
        optimizer.step()

    return np.mean(loss_list)
```

forward

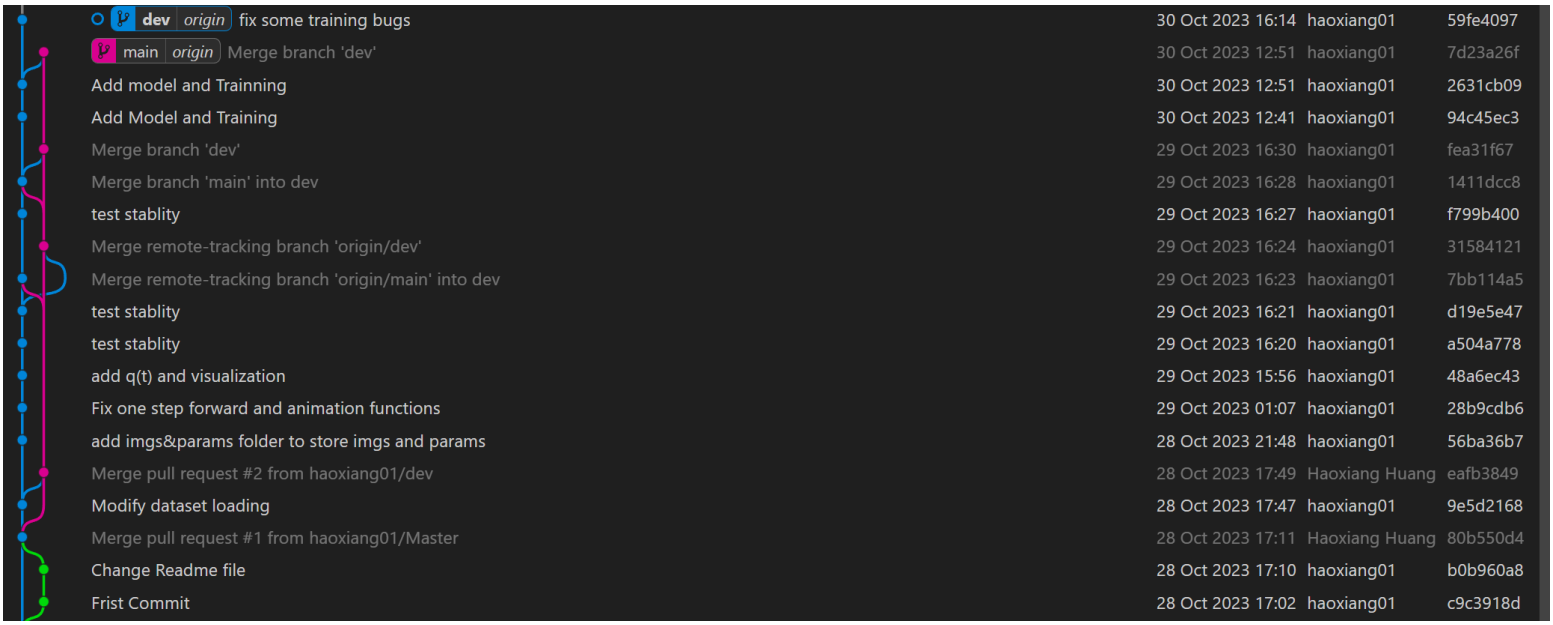
reverse predict

loss = ||εt − εθ(xt, t)||

training visualize:

```
Loss in epoch 3300 is: 0.017955930132804245

<Figure size 600x400 with 0 Axes>
```

Appendix: Project Git record:



dev origin fix some training bugs	30 Oct 2023 16:14	haoxiang01	59fe4097
main origin Merge branch 'dev'	30 Oct 2023 12:51	haoxiang01	7d23a26f
Add model and Trainning	30 Oct 2023 12:51	haoxiang01	2631cb09
Add Model and Training	30 Oct 2023 12:41	haoxiang01	94c45ec3
Merge branch 'dev'	29 Oct 2023 16:30	haoxiang01	fea31f67
Merge branch 'main' into dev	29 Oct 2023 16:28	haoxiang01	1411dcc8
test stablity	29 Oct 2023 16:27	haoxiang01	f799b400
Merge remote-tracking branch 'origin/dev'	29 Oct 2023 16:24	haoxiang01	31584121
Merge remote-tracking branch 'origin/main' into dev	29 Oct 2023 16:23	haoxiang01	7bb114a5
test stablity	29 Oct 2023 16:21	haoxiang01	d19e5e47
test stablity	29 Oct 2023 16:20	haoxiang01	a504a778
add q(t) and visualization	29 Oct 2023 15:56	haoxiang01	48a6ec43
Fix one step forward and animation functions	29 Oct 2023 01:07	haoxiang01	28b9cdb6
add imgs&params folder to store imgs and params	28 Oct 2023 21:48	haoxiang01	56ba36b7
Merge pull request #2 from haoxiang01/dev	28 Oct 2023 17:49	Haoxiang Huang	eafb3849
Modify dataset loading	28 Oct 2023 17:47	haoxiang01	9e5d2168
Merge pull request #1 from haoxiang01/Master	28 Oct 2023 17:11	Haoxiang Huang	80b550d4
Change Readme file	28 Oct 2023 17:10	haoxiang01	b0b960a8
Frist Commit	28 Oct 2023 17:02	haoxiang01	c9c3918d