DDPM 2020

Sunday, June 8, 2025 4:44 AM

DDPM Diffusion Model

High Level Overview

We seek to learn a distribution such that we can generate new images by sampling from this distribution.

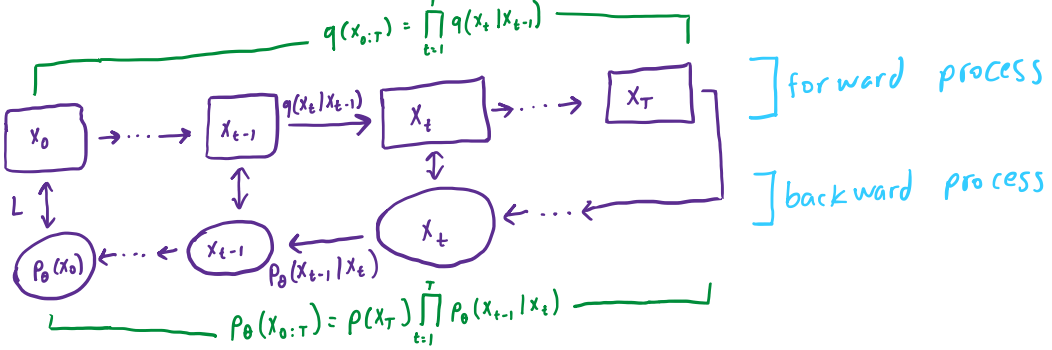
High Level Visualization

Training

Inference

Terminology

- $q(x_0)$ ← Original data distribution
- x_0 ← Real image sampled from q
- x_T ← Pure noise image
- $p(x_T)$ ← Prior Distribution we start with for de-noising : $N(0, I)$ (standard normal distribution)
- $p_\theta(x_0)$ ← Generated denoised image



Forward Process (Adding Noise)

Formula 1: $q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I)$ (mean, variance)

- $q(x_t|x_{t-1})$: conditional probability distribution of a noisy image given a less noisy image
- $N(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I)$: we sample x_t from a Normal distribution, where the mean is the previous (less noisy) image scaled by $\sqrt{1-\beta_t}$, and the variance is $\beta_t I$

Meaning: How likely is it to see a particular noisy image x_t, given that we start from x_{t-1}?

Purpose: Defines how we add noise at each time step

Insight: We are not trying to learn images directly (there are infinitely many) — we are learning the distribution of images. We can then sample images from the learned distribution

Formula 2 (using Formula 1): $q(x_{1:T}|x_0) = \prod_{t=1} q(x_t|x_{t-1})$

- $q(x_{1:T}|x_0)$: joint probability of entire sequence of noisy images
- $\prod_{t=1} q(x_t|x_{t-1})$: product of every conditional probability of a noisy image given it's less noisy image
- $q(x_t|x_{t-1})$: Formula 1

meaning: How likely is this sequence of noisy images $(x_1, x_2, \dots, x_T)$ if we start from x_0 and apply the forward diffusion process?

Purpose: It tells us a probability distribution of the noisy images given a clean image—which lets us generate training data and define our training objective.

Insight: $q(x_{1:T}|x_0)$ = the true distribution. We are trying to estimate p_θ to be as close to q as possible, but we don't know p_θ exactly, because it's too complex. But since we defined the forward process, we know q. So we estimate p_θ by sampling from q, and use the samples from p_θ and q to compute loss.

Forward Process Trick

Instead of adding noise at each timestep, we can add all noise at once.

Given:
- $\alpha_t = 1 - \beta_t$
- $\{\beta_1, \beta_2, \dots, \beta_t\}$

we can compute:

$\{\alpha_1, \alpha_2, \dots, \alpha_T\}$

$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$: cumulative product of each α from 1 to t

we get:

$\{\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_T\}$

we need $\bar{\alpha}_t$ at each timestep for computing loss, but we

Reverse Process (Removing Noise)

Formula 3: $p(x_T) = N(x_T; 0, I)$ (μ, σ^2)

- $p(x_T)$: prior distribution of x_T
- $N(x_T; 0, I)$: Standard Gaussian centered at 0 with unit variance

meaning: we define a prior distribution for the pure noise image, which is just the standard normal distribution

Purpose: we need a starting point for the reverse diffusion process. Since x_T is assumed to be pure noise, we can approximate it as $N(0,1)$. From here, we estimate p_θ

Insight: We need a prior because during inference, we don't have $q(x_T|x_0)$ because we don't have x_0 (this is what we are trying to generate), so we must start with $p(x_T)$, which is our best guess for $q(x_T)$.

Formula 4: $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ (mean, variance)

- $p_\theta(x_{t-1}|x_t)$: conditional probability of a less noisy image given a noisy image
- $N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$: We sample x_{t-1} from a normal distribution:
  - the mean is a learnable parameter and depends on the noisy image and the timestep
  - the variance is a fixed parameter that only depends on the timestep

meaning: How likely is it that we see a less noisy image, x_{t-1}, given a more noisy image, x_t?

Purpose: Tells us how to remove noise from an image

[illegible] probabilities from it during training, where we then

Insight: We are learning a distribution, p_θ, and evaluating probabilities [illegible] compute loss, which then updates p_θ.

Formula 5: $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$

- $p_\theta(x_{0:T})$: joint probability of generating the entire sequence of images
- $p(x_T)$: Formula 3
- $p_\theta(x_{t-1}|x_t)$: Formula 4
- $p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$: prior probability multiplied by the products of the conditional probabilities of less noisy images given their more noisy images

Meaning: Probability of generating this entire sequence starting from x_T

Purpose: We compare this probability value to that of the forward process, enabling us to estimate $p_\theta(x_0)$

Insight: The goal of the diffusion model is to estimate the distribution of the images, and p_θ represents our estimation of the true (unknown) distribution. We can't directly compute $p_\theta(x_0)$, so we start with our best guess for the noise, $p(x_T)$, and remove noise at each time step until we get $p_\theta(x_0)$. When we get $p_\theta(x_0)$, we compare $p_\theta(x_{0:T})$ with the true forward process, $q(x_{1:T}|x_0)$.

Training

We use the above formulas to optimize the negative log likelihood.

Negative Log Likelihood (NLL):

- Loss Function
- If model is confident and correct, NLL is low (good)
- If model is wrong or unsure, NLL is high (bad)

Ex:
We are trying to predict y given x, $p(y|x)$

Likelihood of observing data is $L = \prod_{i=1}^{N} p(y_i|x_i)$ (assuming independence)

Log-Likelihood

$$\log(L) = \sum_{i=1}^{N} \log(p(y_i|x_i)) \quad \text{(base-e)}$$

$$NLL = -\log(L) = -\sum_{i=1}^{N} \log(p(y_i|x_i))$$

- we negate it because we want to minimise loss

Computing NLL for our diffusion process:

$$E[-\log p_\theta(x_0)] \leq E_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] = E_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] = L$$

$p_\theta(x_0)$: our fully denoised image

KL Divergence

- Given:
  - $q(x)$: the true/target distribution
  - $p(x)$: model/predicted distribution

$$D_{KL}(q(x)||p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx$$

- If $p(x)$ is close to $q(x)$, KL divergence is small
- If $p(x)$ is different from $q(x)$, KL divergence is large
- If $p(x) = q(x)$, KL divergence is 0
- Not symmetric

Computing Evidence Lower Bound (ELBO):

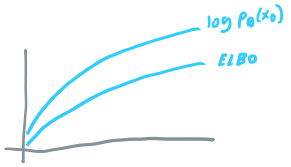
Problem:
we want to maximize $\log p_\theta(x_0)$, but we can't compute this directly.

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T})\, dx_{1:T}$$

$p_\theta(x_{0:T})$: what we are trying to learn

Solution: we compute a lower bound (ELBO):

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



reverse step distribution, which is the joint probability of the (t-1)th noisy state x_{t-1} and the current noisy state x_t given it's (t) noisy state x_t and the clean data, x_0

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

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

$$L = E_q\Big[\underbrace{D_{KL}(q(x_T|x_0)\,||\,p(x_T))}_{L_T} + \sum_{t>0} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0)\,||\,p_\theta(x_{t-1}|x_t))}_{L_{t-1}} \underbrace{- \log p_\theta(x_0|x_1)}_{L_0}\Big]$$

L_T: KL divergence between the distribution that defines adding noise to x_0 and the prior distribution

L_{t-1}: KL divergence between the known reverse step distribution and the reverse step distribution modeled by the neural network

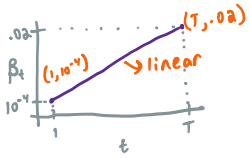
L_0: negative log likelihood of the true data point x_0 under the final reverse step

Purposes:
- L_T: Ensures x_T (noisy image) matches pure gaussian noise
- L_{t-1}: Trains model to denoise $x_t \to x_{t-1}$
  - $\sum L_{t-1}$ represents all timesteps we need
- L_0: Trains model to produce realistic final data $x_1 \to x_0$

Implementation Choices

Forward Choices and L_T

- Authors choose to fix β_t instead of parameterizing them. Instead, they fix them as constants:



Reverse Process and $L_{1:T-1}$

Given $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, we have two choices for $\Sigma_\theta(x_t, t) = \sigma^2 I$, the learnable parameter for the variance given the image at the time step and the time step:

1) $\sigma^2 = \beta_t$ ← optimal if the original image $x_0 \sim N(0, I)$ (the images follow the gaussian distribution)

2) $\sigma^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ ← optimal if the original image x_0 is not gaussian

Both choices have similar results per the paper

The mean, $\mu_\theta(x_t, t)$, needs to be parameterized, and the paper presents the following, given $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$, we can rewrite L_{t-1}:

$$L_{t-1} = E_q\Big[\frac{1}{2\sigma_t^2}\|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|^2\Big] + C$$

$\mu_\theta(x_t, t) \in \mathbb{R}^D$ (D = # of pixels/features)

We learn this vector with the Neural Network

norm of vector difference between the known forward posterior and what the neural network predicts

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

C is a constant that doesn't depend on θ

$$L_{t-1} - C = E_{x_0, \epsilon}\left[\frac{1}{2\sigma_t^2}\left\| \tilde{\mu}_t\left(x_t(x_0,\epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t(x_0,\epsilon) - \sqrt{1-\bar{\alpha}_t}\epsilon)\right) - \mu_\theta(x_t(x_0,\epsilon), t)\right\|^2\right]$$

$$= E_{x_0, \epsilon}\left[\frac{1}{2\sigma_t^2}\left\| \frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(x_t(x_0,\epsilon), t)\right\|^2\right]$$

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

$$E_{x_0, \epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$$

= mean square error between true noise ϵ and neural network's predicted noise ϵ_θ

Data scaling, reverse process decoder, and L_0

We scale pixel values in $\{0, 1, \ldots, 255\}$ linearly to $[-1, 1]$

Problem: $p(x_0|x_1)$ = density, not a probability, so the probability of one point x_0^i is zero.

Solution: We integrate over a small bin around x_0^i

$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} N(x; \mu_\theta^i(x_1, 1), \sigma_1^2)\, dx$$

σ_1^2: our choice for the variance

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + 1/255 & \text{if } x < 1 \end{cases}$$

$$\delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - 1/255 & \text{if } x > -1 \end{cases}$$

$p_\theta(x_0|x_1)$: probability of the clean image x_0 given the noisy image x_1

Probability that pixel i matches x_0^i

"what is the probability that every pixel of the model's Gaussian prediction would land in the correct bin

Meaning: The probability of the fully de-noised image, x_0, given the last noisy image, x_1, is the product of probabilities that each generated pixel matches the true data pixel in x_0 for each pixel.

Purpose: Gives a way to compute L_0 in the loss equation

Paper Summary

- Our data distribution is $q(x_0)$
- We define $q(x_{0:T})$, which is a Markov chain that adds noise to the data
- Our goal is to train $p_\theta(x_{0:T})$, the reverse de-noising process such that we start with pure noise x_T and generate samples x_0 that match the true data distribution $q(x_0)$.
- Training is done by maximizing a lower bound on the data log-likelihood, which encourages each step of the reverse process to accurately reverse the noising process.

Questions / Points of Confusion

- How is the vanishing gradient problem avoided if we are taking the joint probability of thousands of small terms?
- Why do we need to structure the mean we are learning as $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$ instead of just $\mu_\theta(x_t, t) = \gamma$, where the neural network learns γ directly?
- I am still a little unclear on how predicting the probability of a sample or de-noising process, which are both numerical values, helps us