Denoising Dufission Probabilistic Models (DDPM)

Jonathan Ho*, Ajay Jain*, Pieter Abbeel*
Presenter: Pengyu Li†

* UC Berkeley
† University of Michigan, Ann Arbor

Intro

The core of Machine Learning (neural network) sometimes is the quality or quantity of training data. Our model could learn the correct hidden distribution.

1. **real data are hard to obtain** – spectrometer Reconstruction (solving inverse problem using neural network)
2. **data are imbalanced** – multi-class classification (minor class will collapse to major class) (Neural collapse with imbalanced class sample: paper's title is "Imbalanced Trouble")

We need to synthetically generate "good" training data.

Deep generative models

Generative Models: Learning hidden distribution of data and generate novel data which the model never seems before. (Speech, Image, Music, Art).

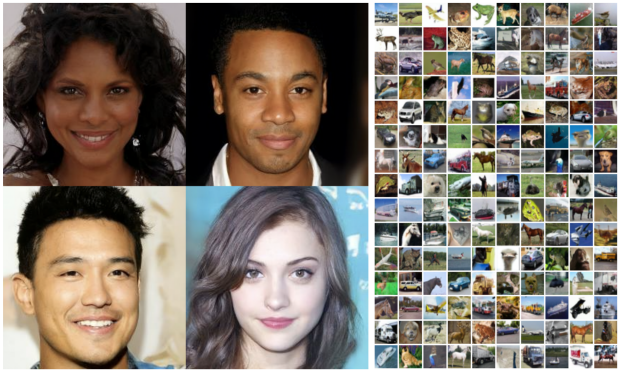

Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Approaches:

Approaches[1] used to synthesize striking image and audio samples but with disadvantage:

- **Generative adversarial networks (GANs)**: the two networks in a GAN (the generator and the discriminator) are constantly competing against others, which can make training unstable and slow.
- **Autoregressive Models:** sensitive to outliers
- **Flow-based** (not time efficient)
- **variational autoencoders (VAEs)** (time efficient but low quality)

[1] more references could be found in the introduction of the DDPM paper

Approaches continued

- **Non-equilibrium Statistical Physics?**
  1. slowly destroy structure in a data distribution
  2. learn a reverse diffusion process that restores structure
  3. tractable generative model of the data
- **Q**: capable of generating high quality samples?
- **DDPM:** Yes! Also denoising.

Non-equilibrium Statistical Physics = Diffusion Model

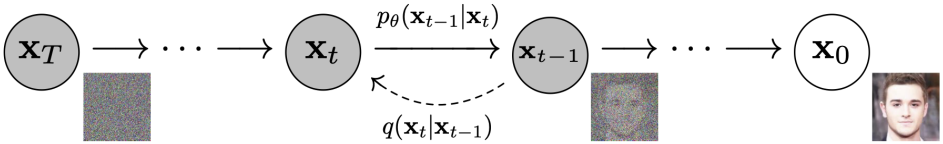

Figure: Demonstration of Diffusion model

Structure:

1. Parameterized Markov Chain (adds noise to destroy signal)
2. Reverse a diffusion process

Function & strength:

1. generating high quality samples
2. denoising
3. straightforward to define and efficient to train

Background

Main Idea: determine a learn-able mapping of data dist. from its prior dist.(adding noisy) and then learn adjoint/reversed mapping from prior dist. back to data dist. (*Forward* and *Reverse*).

Define:

- $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T$ ($\boldsymbol{x}_{1:T}$) are latent variables with same dim as data $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
- ***Forward*** process: gradually adding noise in a Markov chain fashion (posterior $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$)
- ***Reverse*** Process: gradually converting noisy to "clean" data similarly (joint distribution $p_\theta(\boldsymbol{x}_{0:T})$)

Background - math

Forward:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

1. generate “noisier” given “cleaner” with known/given mean and variance
2. fixed MC with Gaussian noise
3. variance schedule $\beta_1, \cdots, \beta_T$ are very small (gradual diffusion)
4. β_t’s are learn-able by reparametrization or just very small constant

Background - math

Reverse:

$$p_\theta(\mathbf{x}_{0:T}) \coloneqq p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

1. estimate “cleaner” given “noisier” with unknown mean and variance (We could estimate it with network parameters θ)
2. starting point is completely noise: $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; , \mathbf{0}, \mathbf{I})$
3. by definition, it is a MC with learned Gaussian transition

Forward vs. Reverse

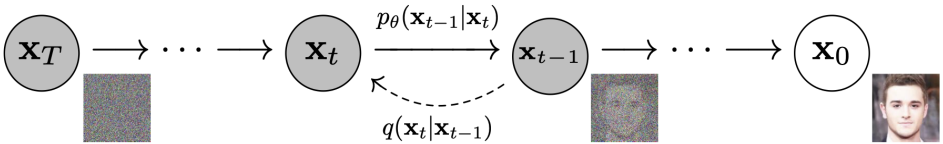

Figure: Demonstration of Diffusion model

- q: Forward, adding noise as t increase
- p_θ: Reverse, denoising

Forward-continued

Forward process admits sampling $\boldsymbol{x}_t$ at an arbitrary time-step t in a closed form directly from the initial given signal $\boldsymbol{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

- $\alpha_t := 1 - \beta_t$ and $\hat{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- given $\boldsymbol{x}_0$, easy to compute $\boldsymbol{x}_t$
- could be used to simplify the forward process posteriors ($q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$)

Reverse-continued

Now that we have defined the forward chain $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, we can theoretically compute the reverse chain $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ when conditioned on $\boldsymbol{x}_0$:

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

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

since by **Bayes's rule**

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}$$

Challenge

The reverse (denoising chain which is Gaussian) could be calculate as:

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

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Challenge: when doing denoising, we don't have access to the original signal/image $\boldsymbol{x}_0$.

Solution: learn a neural network with parameter θ that estimate $\boldsymbol{x}_0$, given $\boldsymbol{x}_t$ (noisy image) and t (noise level). i.e. we want $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to be close to $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$

Objective function

Inspired by the variational bound:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L$$

LHS is negative log likelihood measure how well the model fit the training data, which is hard to maximize. Instead the authors could maximize the lower bound on the RHS.

We can rewrite the RHS L in terms of KL divergence that compares relative entropy of two distributions.

$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big]$$

Object continued

$$\mathbb{E}_q \Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Bigg]$$

- L_T: could be calculated directly for given β_t's. (forward process)
- L_{t-1}: (reverse process) still with unknown reverse chain $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$
- L_0: $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)$ is the last term of the reverse process (adjustable decoder)

Objective function continued

$$\sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

Objective function continued

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

After more reparametrization of $\boldsymbol{x}_t(\boldsymbol{x}_0, \epsilon)$ where, ϵ is simply standard Gaussian, we can rewrite $L_{t-1} - C$ as

$$= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t) \right\|^2 \right]$$

The μ_θ must predict or converge to $\frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right)$ for $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Objective function continued

The μ_θ converges to/predicts $\frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$ for $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In addition, $\boldsymbol{x}_t$ is available so we could further reparametrize μ_θ as:

$$\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

where ϵ_θ is a function approximator intended to predict ϵ from $\boldsymbol{x}_t$.

Recall we defined the reverse process as

$$p_\theta(\mathbf{x}_{0:T}) \coloneqq p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Final Results:

Thus, to sample/predict x_{t-1} is to

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$$

where ϵ_θ could be learned as a gradient of reparametrized

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right]$$

Algorithm

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** $\mathbf{x}_0$

- Algorithm 1 is using to predict the noise level
- Algorithm 2 is then step by step recover/sample $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$ and so on to eventually get $\boldsymbol{x}_0$

Denoising Experiment Results

Figure 6: Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right). Extended samples and sample quality metrics over time in the appendix (Figs. 10 and 14).

Gnerative Experiment Results

Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.

Frame Title

1. For the DDPM, should we choose β_t large or small?

2. Is the network trained to estimate forward or reverse chain? If so why we don't need to train on the other? This is also why DDPM is relatively more straightforward and efficient to train.)