Unsupervised Learning

CS446-Machine Learning
CS444-Deep Learning for Computer Vision

Qi Long

Contents

0 Preliminaries

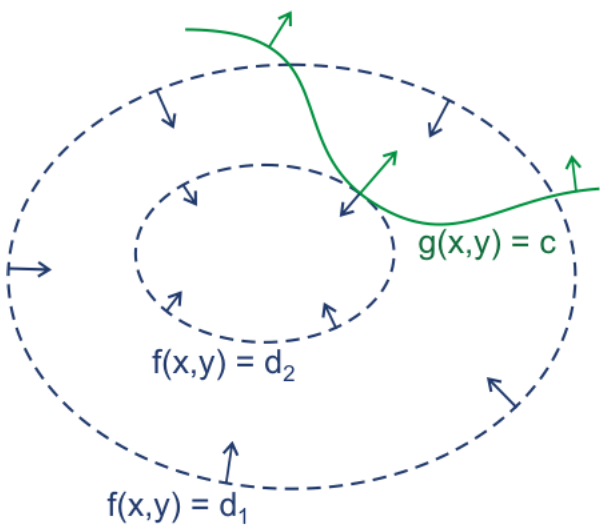
0.1 Lagrangian Multiplier

0.1.1 Goal: Find Optimum

Find the optimum of a function given a restriction of another function, for example,

$$\max_{g(x,y)=c} f(x,y)$$

0.1.2 Method: same derivative



$f(x,y) = d_n$ can be regarded as contours with **adjustable** values d_n.
$g(x,y) = c$ can be regarded as a fixed curve crossing contours.
Then the optimum is when $f(x,y) = d_n$ and $g(x,y) = c$ have same derivative.
Import λ to represent adjustable d_n,

$$\nabla \frac{1}{\lambda} f(x,y) = \nabla (g(x,y) - c)$$

$$\nabla [f(x,y) - \lambda(g(x,y) - c)] = 0$$

Solve this gives λ, plug back to objective to solve optimum.

0.2 Linear Algebra Basics

$$A(B+C) = AB + AC$$

$$(A+B)C = AC + BC$$

0.2.1 Inverse

$$(ABC\cdots)^{-1} = \cdots C^{-1}B^{-1}A^{-1}$$
$$(A^T)^{-1} = (A^{-1})^T$$
$$(A+B)^T = A^T + B^T$$
$$(AB)^T = B^T A^T$$

0.2.2 Trace

$$Tr(ABC) = Tr(BCA) = Tr(CAB)$$
$$Tr(A+B) = Tr(A) + Tr(B)$$

0.3 Matrix Derivative

0.3.1 First Order

$$\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a$$
$$\frac{\partial a^T X b}{\partial X} = ab^T$$

0.3.2 Second Order

$$\frac{\partial x^T B x}{\partial x} = (B + B^T)x$$
$$\frac{\partial b^T X^T X c}{\partial X} = X(bc^T + cb^T)$$

0.4 Norm

0.4.1 Vector Norm

General **p-norm**:

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p} \qquad , p \geq 1$$

2-norm or **Euclidean norm**: Euclidean distance,

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$||x||_2^2 = x^T x$$

1-norm: Manhattan distance, grid path length,

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

max-norm: Max-dim distance,

$$||x||_\infty = \max_i |x_i|$$

0.4.2 Matrix Norm

Frobenius Norm: sum of squared each entry,

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} = \sqrt{tr(A^T A)}$$

0.5 SVD

For matrix M of size $(m \times n)$,

$$M = U \Sigma V^T$$

0.5.1 Unitary Matrix

$$U^{-1} = U^T$$

$$U^* U = U U^* = I$$

0.5.2 Singular Value Decomposition

- U: size of $(m \times m)$, unitary matrix.
- Σ: size of $(m \times n)$, diagonal matrix uniquely defined by M, diagonal entries are **singular values** of M.
- V: size of $(n \times n)$, unitary matrix.

0.6 Gaussian Distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

0.7 Covariance

$$Cov[X,Y] = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

$$\Sigma = \begin{pmatrix} Var[X_1] & Cov[X_1, X_2] & \cdots & Cov[X_1, X_D] \\ Cov[X_2, X_1] & Var[X_2] & \cdots & Cov[X_2, X_D] \\ \vdots & \vdots & \ddots & \vdots \\ Cov[X_D, X_1] & Cov[X_D, X_2] & \cdots & Var[X_D] \end{pmatrix}$$

0.8 Conditional Probability

0.8.1 Markov Chain

If X, Y, Z form a Markov Chain ($X \to Y \to Z$), then

$$p(x, y, z) = p(x)p(y|x)p(z|y)$$

0.8.2 Bayes Theorem

$$p(x|y, z) = p(y|x, z)\frac{p(x|z)}{p(y|z)}$$

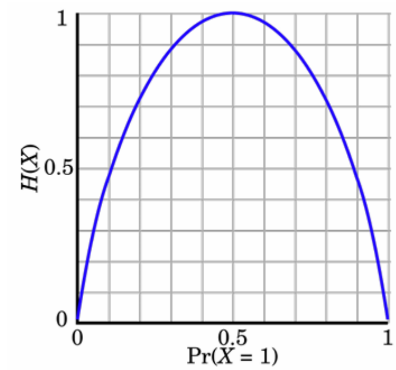
0.9 Information Theory

0.9.1 Shannon Entropy

Higher value corresponds to more uncertainty.

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i$$

$$H(X) = -\sum_{k=1}^{K} p(X = k) \log p(X = k) = -\mathbb{E}[\log p(X)]$$



0.9.2 Cross Entropy

General case:

$$H_{ce}(p,q) = -\sum_{k=1}^{K} p_k \log q_k$$

Binary case:

$$H_{ce}(p,q) = -p\log q - (1-p)\log(1-q)$$

0.9.3 KL-Divergence

Non-negative Evaluation of similarity between two distributions. Low value corresponds to similar distribution.

$$D_{KL}(P||Q) = \sum_{x\in X} P(x)\log\frac{P(x)}{Q(x)}$$

Bernoulli Distributions:

$$D_{KL}(P||Q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$$

Gaussian Distributions:

$$D_{KL}(P||Q) = \frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0) = \frac{1}{2}||\mu_0-\mu_1||_\Sigma^2$$

*The KL-Divergence of two **identical-variance** Gaussians is just Euclidean distance square between two mean vectors.

0.9.4 TV-Distance

An **event** (contain many x) that has largest difference between distribution P and Q.

$$d_{TV}(P,Q) = \sup_{A \subseteq \mathbb{R}^d} |P(A) - Q(A)|$$

0.9.5 Pinsker's Inequality

$$d_{TV}(\cdot,\cdot) \leq \sqrt{\frac{1}{2} D_{KL}(\cdot||\cdot)}$$

0.9.6 Mutual Information

Reduction in uncertainty given another variable

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = D_{KL}(P(X,Y)||P(X)P(Y))$$

0.10 Jensen's Inequality

For any random variable X and any **concave** function f, it holds that

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

1 PCA

1.1 Goal: Dimension Reduction

Find **a small number of "directions"** (low-dim linear subspace) in input space that explain **variation** in input data; re-represent data by projecting along those directions.

1.2 Preprocessing: Center Data

In order to let variance (1) get rid of mean,

$$\mu = \frac{1}{N} \sum_i x_i$$

$$\bar{X} = X - \mu$$

where $\sum_i x_i$ is sum of each dimension over all data points.
$X \rightarrow$ each data point each dim $-$ mean of that dim over all data points.

1.3 Derivation 1: Direction Maximize Variance

1.3.1 Objective

$$\max_{||w||_2^2=1} Var(w^T\bar{x}) = \max_{||w||_2^2=1} \mathbb{E}[(w^T\bar{x}-0)(\bar{x}^T w-0)] = \max_{||w||_2^2=1} w^T\Sigma w \tag{1}$$

where Σ is covariance matrix,

$$\Sigma = \frac{1}{N}(\bar{x}-0)(\bar{x}^T-0) = \frac{1}{N}\bar{x}\bar{x}^T$$

1.3.2 Lagrangian

$f(x) = w^T\Sigma w$, $g(x) = ||w||_2^2 - 1 = w^T w$, get Lagrangian

$$L(w,\lambda) = w^T\Sigma w - \lambda(w^T w - 1)$$

$$\frac{\partial L(w,\lambda)}{w} = (\Sigma + \Sigma^T)w - \lambda(2w) = 0$$

$$\Sigma w = \lambda w$$

Therefore, λ is eigenvalue of Σ. Plug this back to objective get

$$\max w^T\Sigma w = \max \lambda w^T w = \max \lambda ||w||_2^2 = \max \lambda$$

Therefore, λ is largest eigenvalue of Σ, w is corresponding eigenvector.

1.4 Algorithm 1

1.4.1 Find Lower-dim Directions

Take n highest eigenvalues and eigenvectors $U = [w_1, w_2 \cdots]$.

1.4.2 Compress Data

Project data to linear subspace

$$\hat{x} = U^T(x-\mu) \tag{2}$$

1.4.3 Reconstruct Data

Back projection

$$\tilde{x} = U\hat{x} + \mu \tag{3}$$

1.5 Derivation 2: Direction Minimize Reconstruction Loss

1.5.1 Objective

$$L(W) = \frac{1}{N}\sum_{n=1}^{N} ||x_n - reconstruct(compress(x_n, w))||_2^2$$

Using (2), (3),

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} ||x^{(i)} - ww^T x^{(i)}||_2^2 = \frac{1}{N}||\bar{X} - ww^T\bar{X}||_F^2$$

$$= \frac{1}{N}Tr(((I - ww^T)\bar{X})^T((I - ww^T)\bar{X})) = \frac{1}{N}Tr(\bar{X}\bar{X}^T(I - ww^T)^T(I - ww^T))$$

Since projection $I - ww^T$ has the property

$$(I - ww^T)^T(I - ww^T) = I - 2ww^T + w(w^Tw)w^T = I - 2ww^T + ||w||_2^2ww^T = I - ww^T$$

$$L(W) = \frac{1}{N}Tr(\bar{X}\bar{X}^T(I - ww^T)^T(I - ww^T)) = Tr(\Sigma(I - ww^T)) = Tr(\Sigma) - Tr(\Sigma ww^T))$$

because $Tr(\Sigma)$ is fixed wrt w, objective becomes

$$\max_{||w||_2^2=1} w^T\Sigma w$$

Then the rest of derivation is the same as Derivation 1-Lagrangian.

1.6 Solve by SVD

To compute

$$\Sigma = \frac{1}{N}\bar{X}\bar{X}^T$$

let

$$\frac{1}{\sqrt{N}}\bar{X} = USV^T$$

then

$$\Sigma U = USV^TVSU^TU = S^2U$$

Therefore, need to compute Singular Values S and U.

2 Clustering

2.1 K-Means

2.1.1 Objective

Assign each point to a cluster, minimize the total Euclidean distance of data points to clusters' means.

$$\min_{\mu} \min_{r} \sum_{i \in D} \sum_{k=1}^{K} \frac{1}{2} r_{ik} ||x^{(i)} - \mu_k||_2^2$$

where D is dataset, K is number of clusters, r_i is one-hot ($r_{ik} \in 0, 1$, $\sum_{k=1}^{K} r_{ik} = 1$).

2.1.2 Alternate Optimization

Given μ fixed, update r (assign each point to nearest cluster mean, $O(KNd)$),

$$r_{ik} = \begin{cases} 1, & k = \underset{k \in 1, \cdots, K}{\operatorname{argmin}} ||x^{(i)} - \mu_k||_2^2 \\ 0, & otherwise \end{cases}$$

Given r fixed, update μ (recalculate mean of each cluster, $O(Nd)$),

$$\nabla_{\mu_k} L = \sum_{i \in D} r_{ik}(x^{(i)} - \mu_k) = 0$$

$$\mu_k = \frac{\sum_{i \in D} r_{ik} x^{(i)}}{\sum_{i \in D} r_{ik}}$$

Iterate until convergence (no more update).

2.1.3 K-Means++

Initialization: randomly choose first center, pick new centers with

$$p \propto ||x^{(i)} - \mu_k||_2^2$$

(farthest from previous centers).

2.2 Gaussian Mixture Models

2.2.1 Single Gaussian

Goal: use a Gaussian to approximate the data distribution.
By tuning parameters μ, σ, maximize **generative** objective:

$$p(x^{(i)}|\mu, \sigma) = N(x^{(i)}|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x^{(i)} - \mu)^2)$$

minimize negative log likelihood:

$$L = -\log \prod_{i \in D} p(x^{(i)}|\mu, \sigma) = \frac{N}{2} \log(2\pi\sigma^2) + \sum_{i \in D} \frac{1}{2\sigma^2}(x^{(i)} - \mu)^2$$

$$\frac{\partial L}{\partial \mu} = \sum_{i \in D} \frac{-1}{\sigma^2}(x^{(i)} - \mu) = 0 \quad \rightarrow \quad \mu^* = \frac{1}{N} \sum_{i \in D} x^{(i)}$$

$$\frac{\partial L}{\partial \sigma} = \sum_{i \in D} \frac{N}{2} \cdot \frac{4\pi\sigma}{2\pi\sigma^2} - \frac{1}{\sigma^3}(x^{(i)} - \mu)^2 = 0 \quad \rightarrow \quad \sigma^{2*} = \frac{1}{N} \sum_{i \in D} (x^{(i)} - \mu)^2$$

2.2.2 Mixed Gaussian

Soft version of K-Means:

- Each sample is partially assigned to **all** clusters (with responsibility r_{ik}, weight π_k is shared among samples).

$$p(x^{(i)}|\pi, \mu, \sigma) = \sum_{k=1}^{K} \pi_k N(x^{(i)}|\mu_k, \sigma_k)$$

$$r_{ik} = \frac{\pi_k N(x^{(i)}|\mu_k, \sigma_k)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} N(x^{(i)}|\mu_{\hat{k}}, \sigma_{\hat{k}})}$$

- Each cluster's mean is updated biased wrt to samples. (Each cluster is a Gaussian that only takes partial consideration (r_{ik}) of each sample)

$$N_k = \sum_{i \in D} r_{ik}$$

Minimize objective by adjusting parameters π, μ, σ:

$$L = -\log \prod_{i \in D} p(x^{(i)}|\pi, \mu, \sigma) = -\sum_{i \in D} \log \sum_{k=1}^{K} \pi_k N(x^{(i)}|\mu_k, \sigma_k)$$

where π_k is the assigned partial of sample to cluster k, $\sum_{k=1}^{K} \pi_k = 1$.
Similar to single Gaussian except small modifications,

$$\frac{\partial L}{\partial \mu_k} = \sum_{i \in D} r_{ik} \frac{-1}{\sigma^2}(x^{(i)} - \mu) = 0 \quad \rightarrow \quad \mu_k^* = \frac{1}{N_k} \sum_{i \in D} r_{ik} x^{(i)}$$

$$\frac{\partial L}{\partial \sigma_k} = \sum_{i \in D} r_{ik}(\frac{N}{2} \cdot \frac{4\pi\sigma}{2\pi\sigma^2} - \frac{1}{\sigma^3}(x^{(i)} - \mu)^2) = 0 \quad \rightarrow \quad \sigma_k^{2*} = \frac{1}{N_k} \sum_{i \in D} r_{ik}(x^{(i)} - \mu)^2$$

For updating π_k, use Lagrangian,

$$\frac{\partial L}{\partial \pi_k} = \sum_{i \in D} \frac{N(x^{(i)}|\mu_k, \sigma_k)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} N(x^{(i)}|\mu_{\hat{k}}, \sigma_{\hat{k}})} + \lambda = 0$$

$$\sum_{i \in D} \frac{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} N(x^{(i)}|\mu_{\hat{k}}, \sigma_{\hat{k}})}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} N(x^{(i)}|\mu_{\hat{k}}, \sigma_{\hat{k}})} + \sum_{k=1}^{K} \pi_k \lambda = 0$$

$$N + \lambda = 0 \quad \rightarrow \quad \lambda = -N, \pi_k = \frac{N_k}{N}$$

2.3 Evaluation

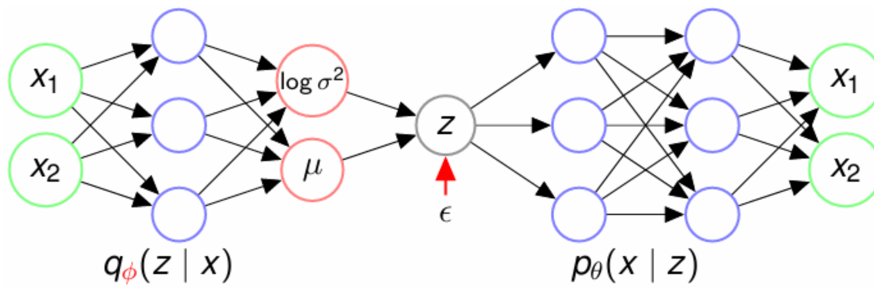
- Reconstruction Loss.
- Purity (if label available).

3 VAE

3.1 Setup

- Encoder: deep net to predict mean and var based on input $x^{(i)}$, so for a different $x^{(i)}$ will get a different Gaussian.

$$q_\phi(z|x) = N(z; \mu_\phi(x), \sigma_\phi(x))$$



- Sample latent var: parameterization trick

$$z \sim q_\phi(z|x) = N(z; \mu_\phi(x), \sigma_\phi(x)) = \mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon$$

  where $\epsilon \sim N(0, 1)$

- Decoder: deep net to predict x from latent var. During Inference time, only decoder is used (with a sample from Normal Distribution as input).

$$\hat{x} = p_\theta(x|z)$$

3.2 ELBO

3.2.1 From Objective to ELBO

Objective is to maximize the probability of decoder generating ground truth x.
Marginalize all latent var z:

$$\log p_\theta(x) = \log \int p_\theta(x, z) dz = \log \int q_\phi(z|x) \frac{p_\theta(x, z)}{q_\phi(z|x)} dz$$

By Jensen-inequality:

$$\geq \int q_\phi(z|x) \log \frac{p_\theta(x, z)}{q_\phi(z|x)} dz = L(p_\theta, q_\phi)$$

3.2.2 From ELBO to Loss

The term $p_\theta(x, z)$ is intractable, use bayes to separate it into two tractable losses:

$$L(p_\theta, q_\phi) = \int q_\phi(z|x) \log \frac{p_\theta(x, z)}{q_\phi(z|x)} dz = \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} dz$$

$$= \int q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz + \int q_\phi(z|x) \log p_\theta(x|z) dz = -D_{KL}(q_\phi, p) + \mathbb{E}[\log p_\theta(x|z)]$$

Interpretation of loss:

- Prior Matching (Regularization): restrict $q_\phi(z|x)$ close to $N(z;0,1)$ ($p(z)$) to avoid encoder directly copying input instead of discovering features.

$$-D_{KL}(q_\phi||p) = \int q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz$$

- Reconstruction: minimize $(x - decode(encode(x)))$.

$$\mathbb{E}[\log p_\theta(x|z)] = \int q_\phi(z|x) \log p_\theta(x|z) dz$$

3.2.3 Methodology

From Objective to ELBO: transform $p_\theta(x)$ to condition on $q_\phi(z|x)$.

- Goal: log probability of predicting real data. (**Fixed**)

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

- Marginalize all hidden / latent variables.

$$\log p_\theta(x_0) = \log \int p_\theta(x_0, z_{1:T}) dz$$

- Choose q_ϕ to represent **whole** forward path (deriving all hidden / latent variables).

$$q_\phi(z_{1:T}|x)$$

- Extract q_ϕ and apply Jensen-inequality to get expectation over latent space.

$$\int q_\phi(z_{1:T}|x) \log \frac{p_\theta(x_0, z_{1:T})}{q_\phi(z_{1:T}|x)} dz = \mathbb{E}_{z_{1:T} \sim q_\phi(z_{1:T}|x)} \log \frac{p_\theta(x_0, z_{1:T})}{q_\phi(z_{1:T}|x)}$$

From ELBO to Loss: break down ELBO to tractable terms.

- Prior Matching: restrict latent space to Normal Distribution (**latent space**).

$$-D_{KL}(q_\phi(z_T|x_0), N(0,I)) = \int q_\phi(z_T|x_0) \log \frac{N(0,I)}{q_\phi(z_T|x_0)} dz$$

- Reconstruction: evaluate difference between generated data and real data (**data space**).

$$\mathbb{E}[\log p_\theta(x_0|z_1)] = \int q_\phi(z_1|x_0)\log p_\theta(x_0|z_1)dz$$

- Transition Quality*: only for gradual noise injection. (See Diffusion Model)

3.3 Training

The overall training loss:

$$\underset{\phi,\theta}{\text{argmax}} -D_{KL}(q_\phi||p) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

Approximate expectation of reconstruction loss by Monte-Carlo simulation:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] = \frac{1}{N}\sum_{i=1}^{N}\log p_\theta(x^{(i)}|z^{(i)}), \quad z^{(i)} \sim q_\phi(z|x^{(i)})$$

Prior matching loss is a comparison between two Gaussian distributions,

$$D_{KL}(q_\phi(z|x^{(i)})||p(z)) = \frac{1}{2}((\sigma_\phi^2(x^{(i)}))^d + \mu_\phi(x^{(i)})^T\mu_\phi(x^{(i)}) - d\log(\sigma_\phi^2(x^{(i)})))$$

Therefore, total loss can be back-propagated through encoder-decoder network.
(See Github:CS446-MP4 for more details)

4 GAN

4.1 Binary Classification Game

Setup: $\eta(x) = Pr(guessP|x)$.

$$Pr(error|x) = Pr(guessP, x \sim Q) + Pr(guessQ, x \sim P) = \sum_{x\in X}\frac{1}{2}Q(x)\eta(x) + \frac{1}{2}P(x)(1-\eta(x))$$

$$= \frac{1}{2} + \frac{1}{2}\sum_{x\in X}\eta(x)(Q(x)-P(x)) = \frac{1}{2} + \frac{1}{2}\sum_{P(x)\geq Q(x)}Q(x)-P(x) = \frac{1}{2} - \frac{1}{2}d_{TV}(P,Q)$$

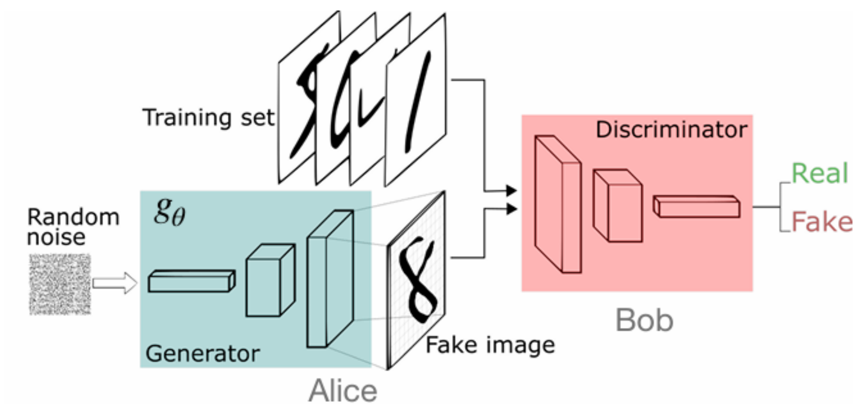
when taking optimal policy

$$\eta(x) = \begin{cases} 1, & P(x) \geq Q(x) \\ 0, & P(x) < Q(x) \end{cases}$$

Similarly, if using Cross Entropy Loss, optimal:

$$\eta(x) = \frac{P(x)}{P(x) + Q(x)}$$

4.2 Non-saturating GAN Loss (NSGAN)



$$\min_\theta \max_\phi V(g_\theta, f_\phi) = \mathbb{E}_{x\sim P_d}[\log f_\phi(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - f_\phi(g_\theta(z)))]$$

Interpretation:

- Ground Truth: True data labeled 1, generated data labeled 0.
- Generator: learn to sample from the distribution represented by the training set, try to fool Discriminator.

$$\log(1 - f_\phi(g_{\theta(z)})) \to 1$$

$$D^* = \arg\max_D V(G, D)$$

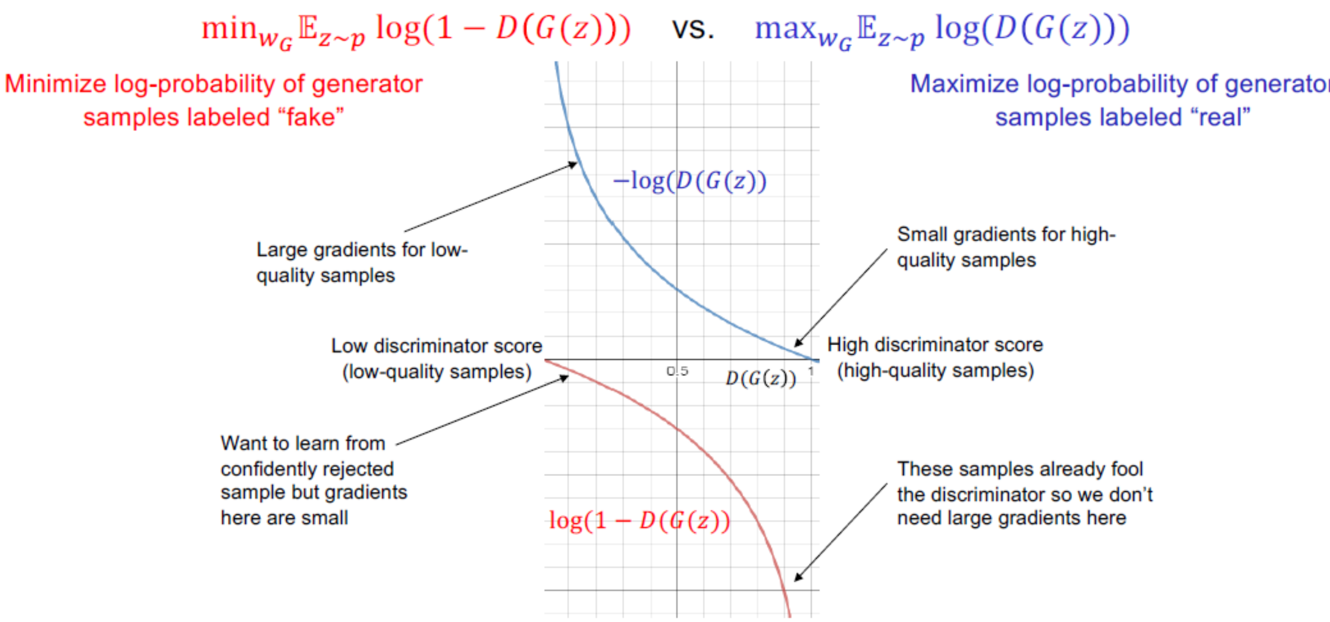
- Discriminator: learn to distinguish between generated and real samples.

$$\log f_\phi(x_d) \to 1$$

$$\log(1 - f_\phi(g_{\theta(z)})) \to 0$$

$$G^* = \arg\min_G V(G, D) = \arg\min_G \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))] = \arg\max_G \mathbb{E}_{z\sim P_z(z)}[\log(D(G(z)))]$$

Reason for max: larger loss focusing on low quality samples.



Limitations: training stability, behavior sensitive to hyperparameter selection. Low quality generations.

4.3 Optimization

Gradient descent-ascent algorithm (asynchronous version) with learning rate γ: Generator descent:

$$\nabla_\theta^{(t)} = \nabla_\theta V(g_\theta^{(t)}, f_\phi^{(t)})$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \nabla_\theta^{(t)}$$

Discriminator ascent:

$$\nabla_\phi^{(t)} = \nabla_\phi V(g_\theta^{(t+1)}, f_\phi^{(t)})$$

$$\phi^{(t+1)} \leftarrow \phi^{(t)} - \gamma \nabla_\phi^{(t)}$$

Similar for synchronous version.

(See Github:CS446-MP5.1 for more details)

4.4 Advanced Architectures

4.4.1 DCGAN

Improve discriminator architecture to reach empirically better training tsability.

- No pooling, only strided convolutions.
- Leaky ReLU activations.
- Only one FC layer before softmax output.
- Batch normalization after most layers.

4.4.2 WGAN

Improve on Loss structure to get better gradients and more stable training.

- Replace sigmoid with linear activation in discriminator.
- Drop logs from objective.

$$\min_G \max_D \left[\mathbb{E}_{x\sim p_{data}} D(x) - \mathbb{E}_{z\sim p} D(G(z))\right]$$

- Clip weights to range $[-c, c]$ to ensure smoothness of discriminator.

4.4.3 LSGAN

Improve on Loss structure with least squares cost to get higher-quality images.

$$L_D = \mathbb{E}_{x\sim p_{data}}(D(x)-1)^2 + \mathbb{E}_{z\sim p}(D(G(z)))^2$$

$$L_G = \mathbb{E}_{z\sim p}(D(G(z))-1)^2$$

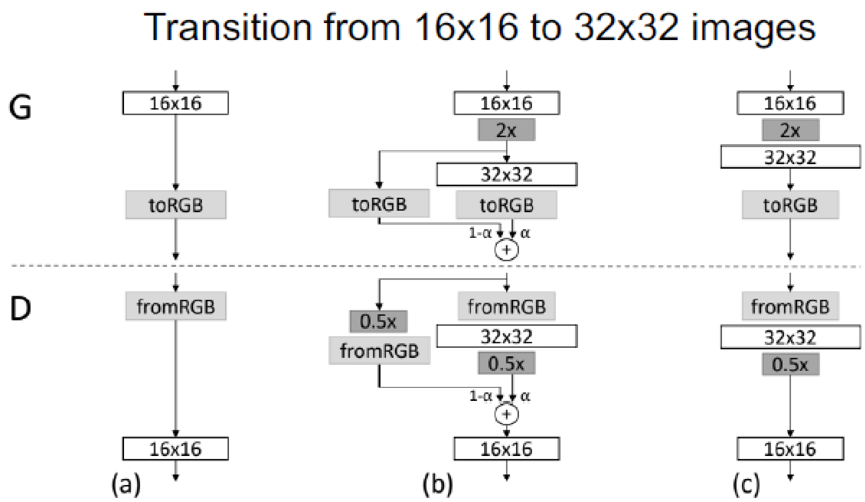
4.4.4 ProgressiveGAN

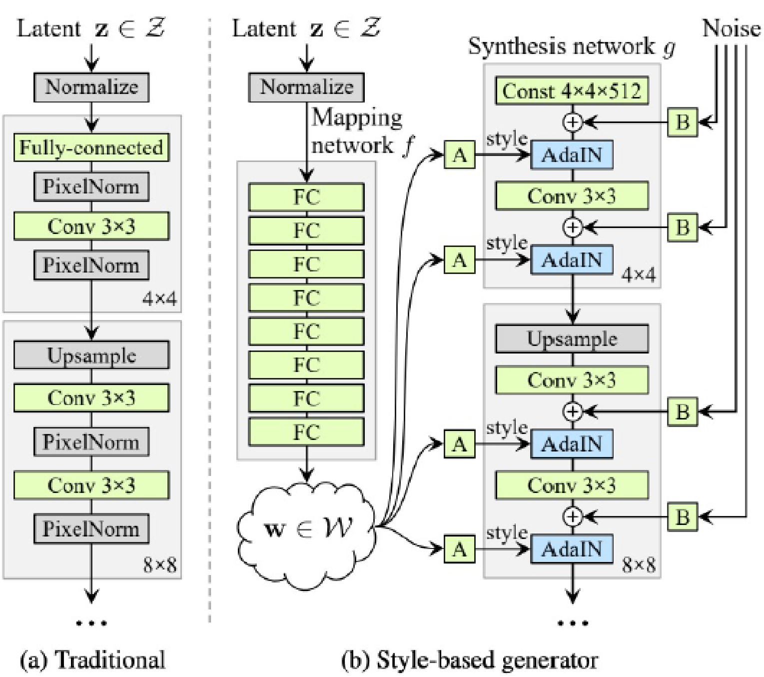
Start from training lower-resolution models, then gradually add layers corresponding to higher-resolution outputs and train.

4.4.5 StyleGAN

Improve based on ProgressiveGAN.

- Start generation with constant instead of noise vector.


Transition from 16x16 to 32x32 images
G
D
16x16
2x
32x32
toRGB
1-α
α
fromRGB
0.5x
(a)
(b)
(c)


Latent $\mathbf{z} \in \mathcal{Z}$
Normalize
Fully-connected
PixelNorm
Conv 3×3
4×4
Upsample
8×8
Mapping network f
FC
$\mathbf{w} \in \mathcal{W}$
Synthesis network g
Noise
Const 4×4×512
A
B
style
AdaIN
(a) Traditional
(b) Style-based generator

- Noise vector is transformed to latent vector w (style codes, control adaptive instance normalization or scaling and biasing of each feature map).
- Add noise after each convolution and before nonlinearity to enable stochastic details.

4.4.6 BigGAN

Scale up self-attention GAN to high resolution images.

4.4.7 StyleGAN-XL

Introduce multiple discriminators operating on projections P_l from a fixed pre-trained feature space.

$$V(G, D) = \sum_l \left(\mathbb{E}_{x \sim p_{data}} \log D_l\left(P_l(x)\right) + \mathbb{E}_{z \sim p} \log\left(1 - D_l\left(P_l(G(z))\right)\right)\right)$$

where each P_l returns a feature map of a different resolution by applying random cross-channel mixing and cross-scale mixing to a pre-trained network.

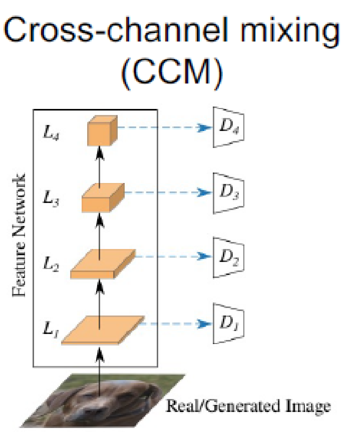


Figure 2: **CCM** (dashed blue arrows) employs 1×1 convolutions with random weights.

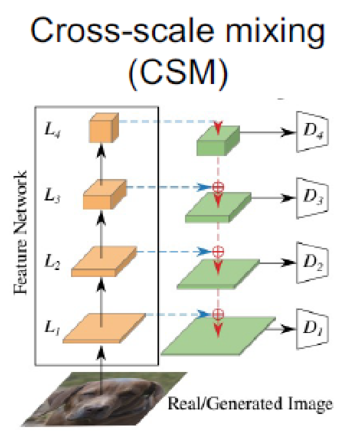


Figure 3: **CSM** (dashed red arrows) adds random 3×3 convolutions and bilinear upsampling, yielding a U-Network.

4.4.8 GigaGAN

Text-to-Image architecture.
Generator: CLIP text encoder embed text to latent code and attention values.
Discriminator: NSGAN, CLIP contrastive loss.

(See Github:CS444-MP4 for more details)

4.5 Evaluation

4.5.1 Human Studies

Turing Tests.

4.5.2 Inception Score (IS)

- Pass generated samples x through an image classifier (InceptionNet), compute posterior class distributions $P(y|x)$ and marginal distribution $P(y)$.
- Compute Inception Score, the higher the better.

$$\mathrm{IS}(G) = \exp\left[\mathbb{E}_{x\sim G}\mathrm{KL}\left(P(y|x)||P(y)\right)\right] = \exp\left[\sum_{x\in G} P(y|x)\log\frac{P(y|x)}{P(y)}\right]$$

  Higher $P(y|x)$ means higher quality that its prediction is closer to ground-truth class. Lower $P(y)$ means generated samples are diverse that contain objects of many classes. So overall the higher IS the better.
  Limitation: can't detect overfitting (memorize training data) or mode-dropping (output a single image per class).

4.5.3 Frechet Inception Distance (FID)

- Pass generated samples x through an image classifier (InceptionNet), compute activations for a chosen layer.
- Estimate multivariate mean and covariance of activations, compute Frechet Inception Distance (FID) for a chosen layer. Limitation: can't detect overfitting (memorize training data) or mode-dropping (output a single image per class).

Can detect mode dropping, but can't detect overfitting.

5 Diffusion

5.1 Setup

Similar to VAE, diffusion model contains an encoder and decoder to approximate the underlying true conditional distribution.

$$\begin{aligned} q_\phi(x_t|x_{t-1}) &\approx p(x_t|x_{t-1}) \\ p_\theta(x_t|x_{t+1}) &\approx p(x_t|x_{t+1}) \end{aligned} \tag{4}$$

- Forward Path: gradually add noise till Gaussian.
- Backward Path: gradually denoise till image output.

Different from VAE, the latent variables are also x_t (same dimension as input image) instead of z.

5.2 Encoder

Manually control the noise infection process so that x_t is deterministic given x_0

$$q_\phi(x_t|x_{t-1}) = N(x_t|\sqrt{\alpha_t}x_{t-1}, (1-\alpha)I)$$

5.2.1 Objective

$$\lim_{t\to\infty} x_t = N(0, I) \tag{5}$$

which is pure Gaussian Noise.

5.2.2 Derivation

Assume

$$x_t = ax_{t-1} + b\epsilon_t$$

where $\epsilon_t \sim N(0, I)$ is pure Gaussian Noise. Then

$$x_t = ax_{t-1} + b\epsilon_t = a(ax_{t-2} + b\epsilon_{t-1}) + b\epsilon_t = a^2x_{t-2} + ab\epsilon_{t-1} + b\epsilon_t$$

$$= \cdots = a^t x_0 + b\sum_{i=0}^{t-1} a^i \epsilon_{t-i} \tag{6}$$

Since ϵ is pure Gaussian Noise $N(0, I)$, $\mathbb{E}[\epsilon_{t-i}] = 0$, $Var(\epsilon_{t-i}) = I$.
Since $a^t x_0$ is a constant, $Var(a^t x_0 + Y) = Var(Y)$.
By linearity of Gaussian, x_t is still a Gaussian Distribution with

$$\mathbb{E}[x_t] = \mathbb{E}[a^t x_0 + b\sum_{i=0}^{t-1} a^i\epsilon_{t-i}] = \mathbb{E}[a^t x_0] + b\sum_{i=0}^{t-1} a^i\mathbb{E}[\epsilon_{t-i}] = a^t x_0$$

$$Var(x_t) = Var(a^t x_0 + b\sum_{i=0}^{t-1} a^i\epsilon_{t-i}) = Var(b\sum_{i=0}^{t-1} a^i\epsilon_{t-i}) = b^2\sum_{i=0}^{t-1} a^{2i} I$$

To make (6) reach objective (5),

$$\begin{cases} \lim_{t\to\infty} a^t x_0 & =0 \\ \lim_{t\to\infty} b^2 \sum_{i=0}^{t-1} a^{2i} & =1 \end{cases} \quad \rightarrow \quad \begin{cases} |a| < & 1 \\ \frac{b^2}{1-a^2} = & 1 \end{cases}$$

Set $a^2 = \alpha_t$, then $a = \sqrt{\alpha_t}$, $b = \sqrt{1-\alpha_t}$,

$$\begin{aligned} x_t &= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon = \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon^{'}) + \sqrt{1-\alpha_t}\epsilon \\ &= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon = \cdots = \sqrt{\prod_{i=1}^{t} \alpha_t} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_t}\epsilon \end{aligned} \tag{7}$$

Therefore, all x_t are tractable given x_0.

5.3 ELBO

Objective is maximizing the log likelihood of decoder predicting real image.

$$\max_\theta \log p_\theta(x_0)$$

Since $p_\theta(x_0)$ itself is intractable, but conditional probability (4) is tractable by decoder, transform objective to marginalize latent variables

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

Since x_t itself is intractable, but the manually controlled encoding process makes it determined by encoder, make $p_\theta(x_{0:T})$ based on output of encoder

$$= \log \int q_\phi(x_{1:T|x_0}) \frac{p_\theta(x_{0:T})}{q_\phi(x_{1:T}|x_0)} dx_{1:T}$$

By Jensen-inequality:

$$\geq \int q_\phi(x_{1:T|x_0}) \log \frac{p_\theta(x_{0:T})}{q_\phi(x_{1:T}|x_0)} dx_{1:T} = \mathbb{E}_{q_\phi(x_{1:T}|x_0)} \log[\frac{p_\theta(x_{0:T})}{q_\phi(x_{1:T}|x_0)}] = ELBO$$

5.4 Disastrous Loss Function

Intuitively, given a latent state x_t, we can compute encoder ($q_\phi(x_t|x_{t-1})$) and decoder ($p_\theta(x_t|x_{t+1})$) predictions from two directions and form a loss compared with the given latent state x_t.
Since encoder and decoder are both one-step conditional prediction, by Markov Chain,

$$ELBO = \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)p_\theta(x_0|x_1)}{q_\phi(x_T|x_{T-1})\prod_{t=1}^{T-1} q_\phi(x_t|x_{t-1})}]$$

$$= \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q_\phi(x_T|x_{T-1})}] + \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=1}^{T-1} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_t|x_{t-1})}]$$

The second term can be further simplified,

$$\mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=1}^{T-1} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_t|x_{t-1})}] = \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(x_{t-1},x_{t+1}|x_0)}[\log \frac{p_\theta(x_t|x_{t+1})}{q_\phi(x_t|x_{t-1})}]$$

Since the distribution $q_\phi(x_{t-1}, x_{t+1}|x_0)$ is not directly tractable, this loss function is not feasible, which is disastrous.

5.5 Reverse Encoder

After a close inspection of this disastrous situation, we can see that the opposite direction of prediction is the core obstacle. Therefore, we can reverse the transition of encoder by Bayes Rule,

$$q_\phi(x_t|x_{t-1}, x_0) = \frac{q_\phi(x_{t-1}|x_t, x_0)q_\phi(x_t|x_0)}{q_\phi(x_{t-1}|x_0)} \tag{8}$$

Since encoder is manually controlled, $q_\phi(x_t|x_0)$ and $q_\phi(x_{t-1}|x_0)$ are both tractable (7), now we consider the remaining term $q_\phi(x_{t-1}|x_t, x_0)$.

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

$$= C_1 \exp(-\frac{1}{2}(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{1-\alpha_t} + \frac{(x_{t-1} - \sqrt{\prod_{i=1}^{t-1}\alpha_t}x_0)^2}{1-\prod_{i=1}^{t-1}\alpha_t} - \frac{(x_t - \sqrt{\prod_{i=1}^{t}\alpha_t}x_0)^2}{1-\prod_{i=1}^{t}\alpha_t}))$$

$$= C_1 \exp(-\frac{1}{2}(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\prod_{i=1}^{t-1}\alpha_t})x_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{1-\alpha_t}x_t + \frac{2\sqrt{\prod_{i=1}^{t-1}\alpha_t}}{1-\prod_{i=1}^{t-1}\alpha_t}x_0)x_{t-1} + C_2)$$

where C_1 and C_2 are constants independent of x_{t-1}, so $\mu = -\frac{B}{2A}$,

$$\mu(x_t, x_0) = (\frac{\sqrt{\alpha_t}}{1-\alpha_t}x_t + \frac{\sqrt{\prod_{i=1}^{t-1}\alpha_t}}{1-\prod_{i=1}^{t-1}\alpha_t}x_0)/(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\prod_{i=1}^{t-1}\alpha_t})$$

$$= \frac{\sqrt{\alpha_t}(1-\prod_{i=1}^{t-1}\alpha_t)}{1-\prod_{i=1}^{t}\alpha_t}x_t + \frac{\sqrt{\prod_{i=1}^{t-1}\alpha_t}(1-\alpha_t)}{1-\prod_{i=1}^{t}\alpha_t}x_0 \tag{9}$$

From (6),

$$x_0 = \frac{1}{\sqrt{\prod_{i=1}^{t}\alpha_t}}(x_t - \sqrt{1-\prod_{i=1}^{t}\alpha_t\epsilon})$$

Plug in x_0,

$$\mu(x_t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\prod_{i=1}^{t}\alpha_t}}\epsilon)$$

Similarly, for variance,

$$\Sigma(x_t) = \frac{(1-\alpha_t)(1-\sqrt{\prod_{i=1}^{t-1}\alpha_t})}{1-\prod_{i=1}^{t}\alpha_t}I$$

Therefore, $q_\phi(x_{t-1}|x_t, x_0)$ is also a Gaussian distribution completely rely on x_t, which has the same direction as decoder.

5.6 Loss Function

Up till now, encoder and decoder can make prediction on the same direction $(x_{t-1}|x_t)$. Deriving ELBO again, this time encoder and decoder prediction should overlap each other,

$$ELBO = \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)\prod_{t=2}^{T}p_\theta(x_{t-1}|x_t)p_\theta(x_0|x_1)}{\prod_{t=2}^{T}q_\phi(x_t|x_{t-1})q_\phi(x_1|x_0)}]$$

$$= \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q_\phi(x_1|x_0)}] + \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=2}^{T}\frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_t|x_{t-1})}] \tag{10}$$

Apply Bayes (8) on product in the second term,

$$\prod_{t=2}^{T}\frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_t|x_{t-1})} = \prod_{t=2}^{T}\frac{p_\theta(x_{t-1}|x_t)}{\frac{q_\phi(x_{t-1}|x_t,x_0)q_\phi(x_t|x_0)}{q_\phi(x_{t-1}|x_0)}} = \prod_{t=2}^{T}\frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t,x_0)} \times \prod_{t=2}^{T}\frac{q_\phi(x_{t-1}|x_0)}{q_\phi(x_t|x_0)}$$

After cancelling same terms on nominator and denominator,

$$\prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_t|x_{t-1})} = \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t, x_0)} \times \frac{q_\phi(x_1|x_0)}{q_\phi(x_T|x_0)}$$

Plug back into ELBO (10), transiting the last term,

$$\begin{aligned}
ELBO &= \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q_\phi(x_1|x_0)}] + \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t, x_0)} + \log \frac{q_\phi(x_1|x_0)}{q_\phi(x_T|x_0)}] \\
&= \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q_\phi(x_1|x_0)} + \log \frac{q_\phi(x_1|x_0)}{q_\phi(x_T|x_0)}] + \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t, x_0)}] \\
&= \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q_\phi(x_T|x_0)}] + \mathbb{E}_{q_\phi(x_{1:T}|x_0)}[\log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t, x_0)}] \\
&= \mathbb{E}_{q_\phi(x_1|x_0)}[\log p_\theta(x_0|x_1)] + \mathbb{E}_{q_\phi(x_T|x_0)}[\log \frac{p(x_T)}{q_\phi(x_T|x_0)}] + \sum_{t=2}^{T} \mathbb{E}_{q_\phi(x_t, x_{t-1}|x_0)}[\log \frac{p_\theta(x_{t-1}|x_t)}{q_\phi(x_{t-1}|x_t, x_0)}] \\
&= \underbrace{\mathbb{E}_{q_\phi(x_1|x_0)}[\log p_\theta(x_0|x_1)]}_{reconstruction} - \underbrace{D_{KL}(q_\phi(x_T|x_0)||p(x_T))}_{prior_match} \\
&\quad \underbrace{- \sum_{t=2}^{T} \mathbb{E}_{q_\phi(x_t|x_0)}[D_{KL}(q_\phi(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))]}_{consistency} = Loss \qquad (11)
\end{aligned}$$

Interpretation of final Loss (11):

- Reconstruction: optimize initial block, generated image is expected to be close to real image.
- Prior Match: optimize final block, $q_\phi(x_T|x_0)$ is expected to be close to $N(0, I)$.
- Consistency: optimize middle transition blocks, generated middle latent variables is expected to be close to that generated by manually controlled encoder.

In a word, diffusion model is a decoder learning from human controlled image noising process. Under the self-supervised learning literature, encoder process is generating data $q_\phi(x_t)$ and label $q_\phi(x_{t-1})$ from original input x_0 while decoder learns from these self-generated data-label pairs in a supervised manner.

5.7 Training

5.7.1 Consistency Loss

Firstly consider the core part of the training, which is minimizing the consistency loss in Loss Function (11). Intuitively it aims at making $q_\phi(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$ as close as possible. We have already derived (9) for $q_\phi(x_{t-1}|x_t, x_0)$ from encoder, since p_θ is a neural network to train and x_t is provided as input for it to predict $p_\theta(x_{t-1}|x_t)$, we can design it in a similar and convenient way.

$$\mu_{p_\theta}(x_t) = \frac{\sqrt{\alpha_t}(1 - \prod_{i=1}^{t-1}\alpha_t)}{1 - \prod_{i=1}^{t}\alpha_t}x_t + \frac{\sqrt{\prod_{i=1}^{t-1}\alpha_t}(1-\alpha_t)}{1 - \prod_{i=1}^{t}\alpha_t}\hat{x_\theta}(x_t) \tag{12}$$

where $\hat{x_\theta}(x_t)$ is the original image prediction by neural network p_θ given x_t.
By MLE, we can now design a training loss that minimizes the difference between means of two Gaussian distributions (9) and (12),

$$L = \frac{1}{2\sigma_q^2(t)}||\underbrace{\mu_q(x_t, x_0)}_{known} - \underbrace{\mu_{p_\theta}(x_t)}_{network}||^2 = \frac{1}{2\sigma_q^2(t)}\frac{\prod_{i=1}^{t-1}\alpha_t(1-\alpha_t)^2}{(1 - \prod_{i=1}^{t}\alpha_t)^2}||\hat{x_{0_\theta}} - x_0||^2$$

5.7.2 Reconstruction Loss

After defining a proper network for decoder, since reconstruction loss (11) also contains decoder network p_θ, we also need to plug in it.

$$\log p_\theta(x_0|x_1) \propto -\frac{1}{2\sigma_q^2(1)}||\mu_\theta(x_1) - x_0||^2$$

$$= -\frac{1}{2\sigma_q^2(1)}||\frac{\sqrt{\alpha_1}(1 - \prod_{i=0}^{0}\alpha_t)}{1 - \prod_{i=0}^{1}\alpha_t}x_1 + \frac{\sqrt{\prod_{i=0}^{0}\alpha_t}(1-\alpha_1)}{1 - \prod_{i=0}^{1}\alpha_t}\hat{x_\theta}(x_1) - x_0||^2$$

Since $\alpha_0 = 1$,

$$= -\frac{1}{2\sigma_q^2(1)}||\hat{x_\theta}(x_1) - x_0||^2$$

5.7.3 Training Algorithm

The prior match loss is guaranteed by manual control over encoder. Combining consistency and reconstruction loss, total loss takes the form:

$$Loss = -\sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{\prod_{i=1}^{t-1} \alpha_t (1-\alpha_t)^2}{(1-\prod_{i=1}^{t} \alpha_t)^2} \mathbb{E}_{q(x_t|x_0)}[||\hat{x_{0_\theta}} - x_0||^2] \tag{13}$$

Then the algorithm is repeating this process:

- Pick a random time stamp $t \sim [1, T]$
- Draw sample from encoder $q_\phi(x_t|x_0)$
- Decoder make prediction $\hat{x_\theta}(x_t)$
- Take gradient descent step on 13

(See Github:CS446-MP5.2 for more details)

5.8 Inference

Same as VAE, inference time only uses decoder. Since for each time stamp, given input latent variable x_t, the prediction of x_{t-1} is input latent variable to stamp $t-1$. Therefore, the output should also be a stochastic sampling (a.k.a VAE). The sampling distribution is a Gaussian with mean predicted according to (12) and variance $\sigma_q^2(t)$.

$$x_{t-1} = \mu_{p_\theta}(x_t) + \sigma_q(t)\epsilon = \frac{\sqrt{\alpha_t}(1-\prod_{i=1}^{t-1}\alpha_t)}{1-\prod_{i=1}^{t}\alpha_t} x_t + \frac{\sqrt{\prod_{i=1}^{t-1}\alpha_t}(1-\alpha_t)}{1-\prod_{i=1}^{t}\alpha_t} \hat{x_\theta}(x_t) + \sigma_q(t)\epsilon \tag{14}$$

where $\epsilon \sim N(0, I)$. Algorithm:

- Input a white noise $x_T \sim N(0, I)$ to decoder
- Decoder make prediction $\hat{x_\theta}(x_t)$
- Sample latent variable for time stamp $t-1$ according to 14
- Repeat until x_0

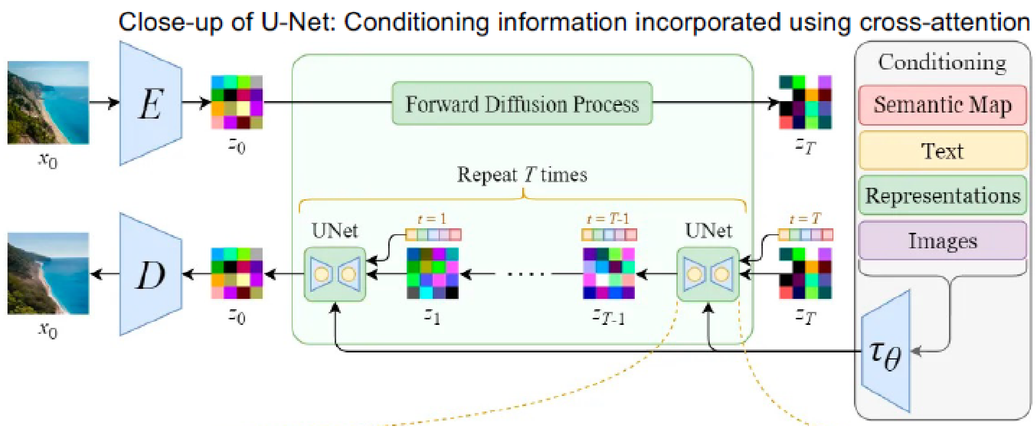
5.9 Advanced Architectures

5.9.1 DALL-E 2

Text-conditioned generation.

- CLIP text encoding: attention mechanism.
- GLIDE diffusion generator: generate image conditioned on CLIP image embedding and text prompt.

5.9.2 Latent Diffusion

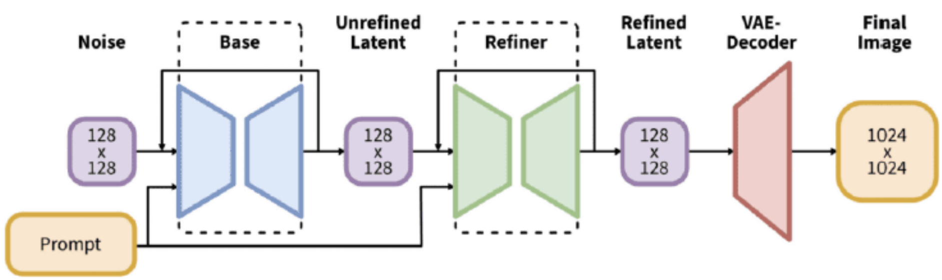


Train a separate encoder and decoder to convert images to and from lower-dimensional latent space, then run conditional diffusion model in latent space instead of original size.

5.9.3 Google Imagen

A frozen LLM text encoder to embed text, a diffusion model to generate images at low resolution and upsamples to higher resolutions.

5.9.4 SDXL



Improve on pipeline, including a Base Diffusion, a Refiner Diffusion and a VAE decoder.

5.9.5 Progressive Distillation

Reducing the intermediate latent space layers progressively.

5.9.6 Latent Consistency Models

Each latent noisy level is used to predict original image and compute loss.