Generative Modelling Overview, GANs and Divergence Minimization

Jayin Khanna
Project Notes: Speech Time-Scale Modification With GANs

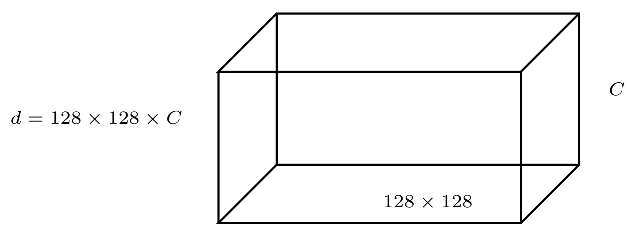
Data and Setup

Let the data be:

$$D = \{x_1, x_2, \ldots, x_n\} \sim \text{iid } P_X \quad \text{(unknown)}$$

where $x_i \in \mathbb{R}^d$, and d is the dimension of the data.

Example: X are images:



Let X_i be a vector-valued random variable of size d. Suppose $D = \{x_1, \ldots, x_n\}$ where $n = 10000$. Then,

$$x_i \sim P_X \quad \text{iid}$$

Generative Modelling

Given: $D = \{x_1, \ldots, x_n\} \sim \text{iid } P_X$

Goal:

1. Estimate P_X
2. Learn to sample from it

General Principle of Generative Models

i) Assume a parametric family on P_X, denoted P_θ.
P_θ: represented using deep neural networks (model)

ii) Define & estimate a divergence (distance notion) metric between P_θ and P_X

iii) Solve an optimization problem over parameters of P_θ to minimize the divergence metric
$\Rightarrow$ to learn the optimal param values from Θ

Example

Let $Z \in \mathbb{R}^k$ be a random variable with some known distribution:

$$Z \sim \mathcal{N}(0, I)$$

Define:

$$g_\theta(z) : Z \to X, \quad \text{then} \quad P_\theta = g_\theta(Z)$$

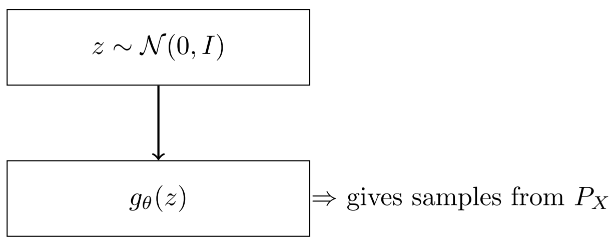
Let $D(P_X \| P_\theta)$ be a divergence measure $\forall P_X, P_\theta$, with:

$$D \geq 0 \quad \& \quad D = 0 \Leftrightarrow P_X = P_\theta$$

Optimization objective:

$$\theta^* = \arg\min_\theta D(P_X \| P_\theta)$$

Once optimized, P_X is estimated implicitly by $g_\theta(z)$. We can sample from P_X using $g_\theta(z)$.



A sample from $z \sim \mathcal{N}(0, I)$, passed through $g_\theta(z)$, produces a sample from $P_\theta = g_\theta(Z)$, which is close to P_X. **Hence, we end up sampling from P_X.**

Important Questions

1. How to compute the divergence metric without knowing P_X and P_θ?
2. What should be the choice of divergence metric?
3. How to choose $g_\theta(z)$ so that in turn $P_\theta \approx P_X$?
4. How to solve the optimization problem of minimizing the divergence?

Each choice gives rise to different methods and tractable sub-problems.

Variational Divergence Minimization

Define divergence metrics between distributions.

1. f-Divergence

Given two probability distributions p_X and p_θ with density functions, define:

$$D_f(p_X \| p_\theta) = \int_{\mathcal{X}} p_\theta(x) f\left(\frac{p_X(x)}{p_\theta(x)}\right) dx$$

Assumptions:

1. Underlying r.v. are continuous
2. Probability distributions have well-defined density

$f : \mathbb{R}_+ \to \mathbb{R}$ satisfying:

- Convex
- Lower semi-continuous
- $f(1) = 0$

Properties of f-Divergence

1. $D_f(p_X \| p_\theta) \geq 0 \quad \forall f$
2. $D_f(p_X \| p_\theta) = 0 \Leftrightarrow p_X = p_\theta$

Common properties:

- Penalizes divergence in the distribution's tails
- Divergences tend to penalize large deviations more severely than small ones

Examples

a) If $f(u) = u \log u$, then it becomes KL-divergence:

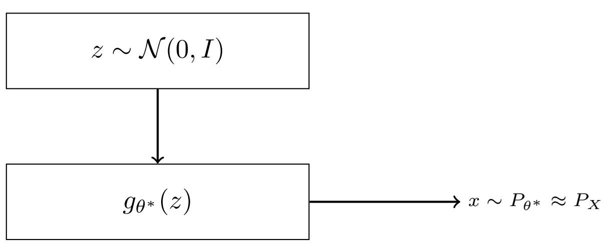
$$D_{\mathrm{KL}}(p_X \| p_\theta) = \int p_X(x) \log \left(\frac{p_X(x)}{p_\theta(x)} \right) dx$$

b) KL-divergence is not symmetric:

$$D_{\mathrm{KL}}(p_X \| p_\theta) \neq D_{\mathrm{KL}}(p_\theta \| p_X)$$

2) $f(u) = \frac{1}{2}\left(u \log u - (u+1)\log(1+u)\right)$: Jensen-Shannon Divergence (JS-divergence)

3) $f(u) = \frac{1}{2}|u-1|$: Total Variation Distance



The above setup becomes a sampler for P_X if $P_\theta \approx P_X$.

Objective

Minimize $D_f(P_X \| P_\theta)$ over parameters θ of g_θ without knowing the exact forms of P_X or P_θ, but only having samples from both.

Key idea: Integrals involving density terms can be approximated using samples drawn from the distributions.

Suppose we want to compute the following integral:

$$I = \int_x h(x) p_X(x) dx$$

where $h(x)$ is a given function and $p_X(x)$ is the (unknown) density.

Since we can draw i.i.d. samples $x_1, \ldots, x_n \sim p_X$, we can approximate:

$$I = \mathbb{E}_{p_X}[h(x)] \approx \frac{1}{n} \sum_{i=1}^{n} h(x_i)$$

using the **Law of Large Numbers** and **Unbiased Monte Carlo estimation**.

Expressing D_f in Expectation Form

Let:

$$h(x) = f\left(\frac{p_X(x)}{p_\theta(x)}\right)$$

Then:

$$D_f(p_X \| p_\theta) = \int_x p_\theta(x) f\left(\frac{p_X(x)}{p_\theta(x)}\right) dx$$

Conjugate Function for Convex f

If $f(u)$ is convex, its Fenchel conjugate is:

$$f^*(t) = \sup_{u \in \text{dom } f} \{ut - f(u)\}$$

A function f is convex if:

$$\forall \lambda \in [0,1],\ f(\lambda u_1 + (1-\lambda)u_2) \leq \lambda f(u_1) + (1-\lambda) f(u_2)$$

Properties of the Conjugate

i) f^* is also convex

ii) $f^{**}(u) = f(u)$ (i.e., f is equal to its biconjugate)

Now:

$$D_f(p_X \| p_\theta) = \int p_\theta(x) f\left(\frac{p_X(x)}{p_\theta(x)}\right) dx = \int p_\theta(x) \sup_t \left\{t \cdot \frac{p_X(x)}{p_\theta(x)} - f^*(t)\right\} dx$$

Interchanging sup and integral (Jensen's inequality):

$$\geq \sup_{T \in \mathcal{T}} \{\mathbb{E}_{p_X}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))]\}$$

Explanation

Define $g(x,t) = \frac{p_X(x)}{p_\theta(x)} t - f^*(t)$. For each x, $\sup_t g(x,t)$ has a definite value $T(x)$ that achieves the maximum.

Thus:

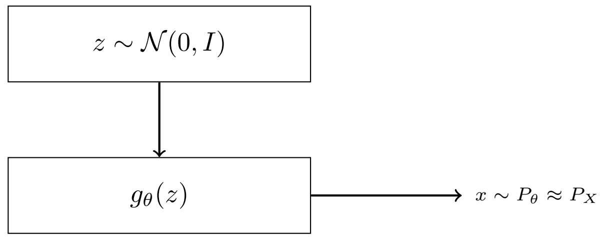
$$D_f(p_X \| p_\theta) \geq \sup_{T(x) \in \mathcal{T}} [\mathbb{E}_{p_X}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))]]$$

This lower bound is tractable using samples from p_X and p_θ.

Realization of VDM (Variational Divergence Minimization)

Given data $D = \{x_1, \ldots, x_n\} \sim p_X$ and prior $z \sim \mathcal{N}(0, I)$, let:

$$P_\theta = g_\theta(z)$$



Let $\mathcal{T}$ be a class of functions $T : \mathcal{X} \to \mathbb{R}$. Then:

$$D_f(p_X \| p_\theta) \geq \sup_{T \in \mathcal{T}} \left[\mathbb{E}_{p_X}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))] \right]$$

We optimize θ and T jointly:

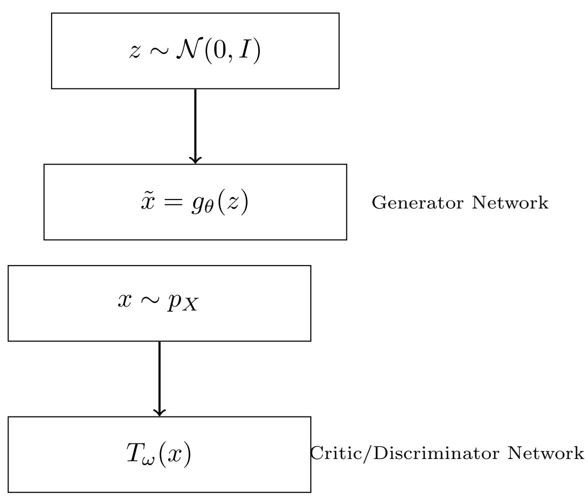
$$\theta^*, T^* = \arg\min_\theta \max_{T \in \mathcal{T}} \left[\mathbb{E}_{p_X}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))] \right]$$

Key Idea: Represent T as a neural network $T_\omega(x)$, where ω are its parameters.
The final objective becomes:

$$\theta^*, \omega^* = \arg\min_\theta \max_\omega \left[\mathbb{E}_{p_X}[T_\omega(x)] - \mathbb{E}_{p_\theta}[f^*(T_\omega(x))] \right]$$

This is the basis for **f-GANs**.

Implementing VDM for Generative Modelling



The loss for Variational Divergence Minimization is:

$$\mathcal{J}(\theta, \omega) = \mathbb{E}_{p_X}[T_\omega(x)] - \mathbb{E}_{p_\theta}[f^*(T_\omega(g_\theta(z)))]$$

Optimization is a saddle-point problem:

$$\theta^*, \omega^* = \arg\min_\theta \max_\omega \mathcal{J}(\theta, \omega)$$

Minimize over generator parameters θ, maximize over critic parameters ω to find equilibrium.

Generative Adversarial Networks (GANs)

For GANs, the f-divergence is chosen as:

$$f(u) = u \log u - (u+1)\log(1+u)$$

This is similar to the Jensen-Shannon divergence.

The Fenchel conjugate is:

$$f^*(t) = -\log(1-e^t), \quad \text{dom } f^* = (-\infty, 0)$$

To ensure $T_\omega(x) \in \text{dom } f^*$, use:

$$T_\omega(x) = \sigma_f(V_\omega(x)) = -\log(1 + e^{-V_\omega(x)})$$

The loss becomes:

$$\mathcal{J}(\theta, \omega) = \mathbb{E}_{p_X}[\sigma_f(V_\omega(x))] - \mathbb{E}_{p_\theta}[f^*(\sigma_f(V_\omega(x)))]$$

Combining all terms:

$$\mathcal{J}_{\text{GAN}}(\theta, \omega) = \mathbb{E}_{p_X}\left[-\log\left(1 + e^{-V_\omega(x)}\right)\right] + \mathbb{E}_{p_\theta}\left[\log\left(1 - \frac{1}{1+e^{-V_\omega(x)}}\right)\right]$$

This reduces to:

$$\mathcal{J}_{\text{GAN}}(\theta, \omega) = \mathbb{E}_{p_X}[\log D_\omega(x)] + \mathbb{E}_{p_\theta}[\log(1 - D_\omega(g_\theta(z)))]$$

where $D_\omega(x)$ is the sigmoid output of the critic.

Implementation of GAN in Practice

Given input data $D = \{x_1, \ldots, x_n\} \sim p_X$:

Update discriminator:

$$\omega^{t+1} = \arg\max_\omega \left[\frac{1}{B_1}\sum_{i=1}^{B_1} \log D_\omega(x_i) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_\omega(g_\theta(z_j)))\right]$$

This step is one gradient ascent step on the discriminator.

Update generator:

$$\theta^{t+1} = \theta^t - \alpha_\theta \nabla_\theta \mathcal{J}_{\text{GAN}}(\theta, \omega)$$

where gradient is passed through g_θ to minimize the generator's loss.

To Train the Discriminator

Keep θ fixed. For a batch $D = \{x_1, \ldots, x_n\}$:

$$\begin{tikzcd}
 & \boxed{z \sim \mathcal{N}(0, I)} \arrow[d] & x \sim p_X \\
 & \boxed{g_\theta(z)} \arrow[d] & \\
\log(1 - D_\omega(g_\theta(z))) & \boxed{D_\omega(x)} & \log D_\omega(x)
\end{tikzcd}$$

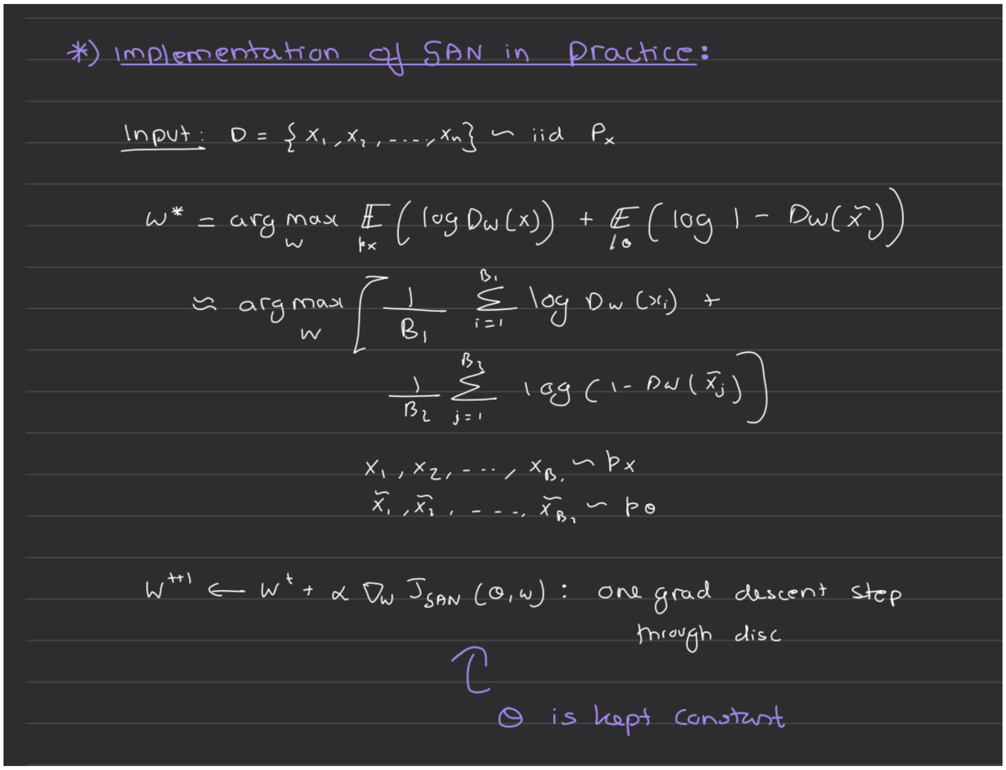


Figure 1: Implementation of GANs in practice from notes

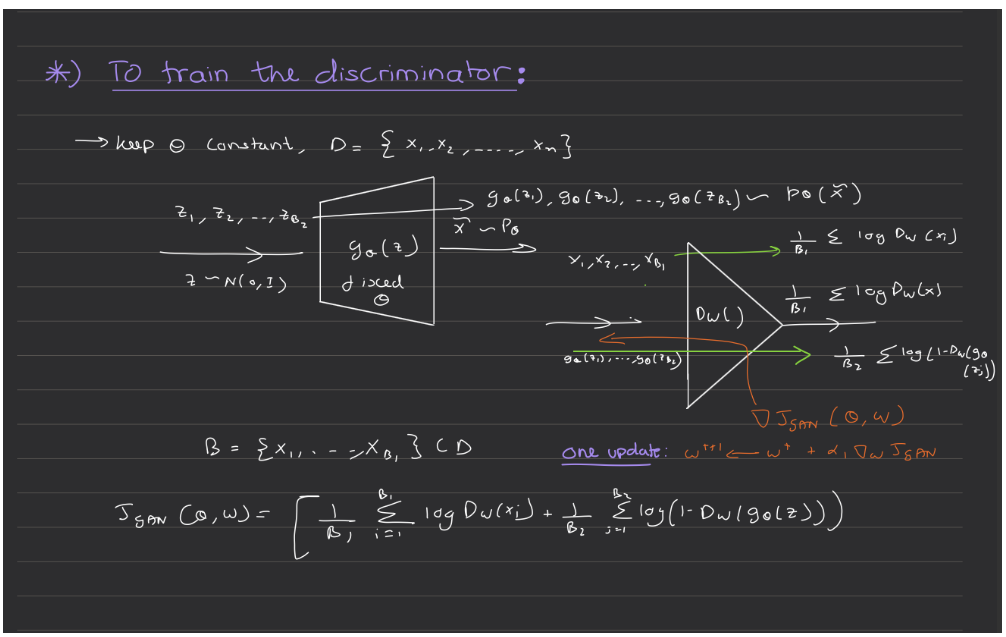


Figure 2: Train Discriminator from notes

Discriminator objective:

$$\mathcal{J}_{\text{GAN}}(\theta, \omega) = \frac{1}{B_1} \sum_{i=1}^{B_1} \log D_\omega(x_i) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_\omega(g_\theta(z_j)))$$

This provides the full formulation and optimization details of Generative Adversarial Networks via f-divergence minimization.

To Train the Generator Network

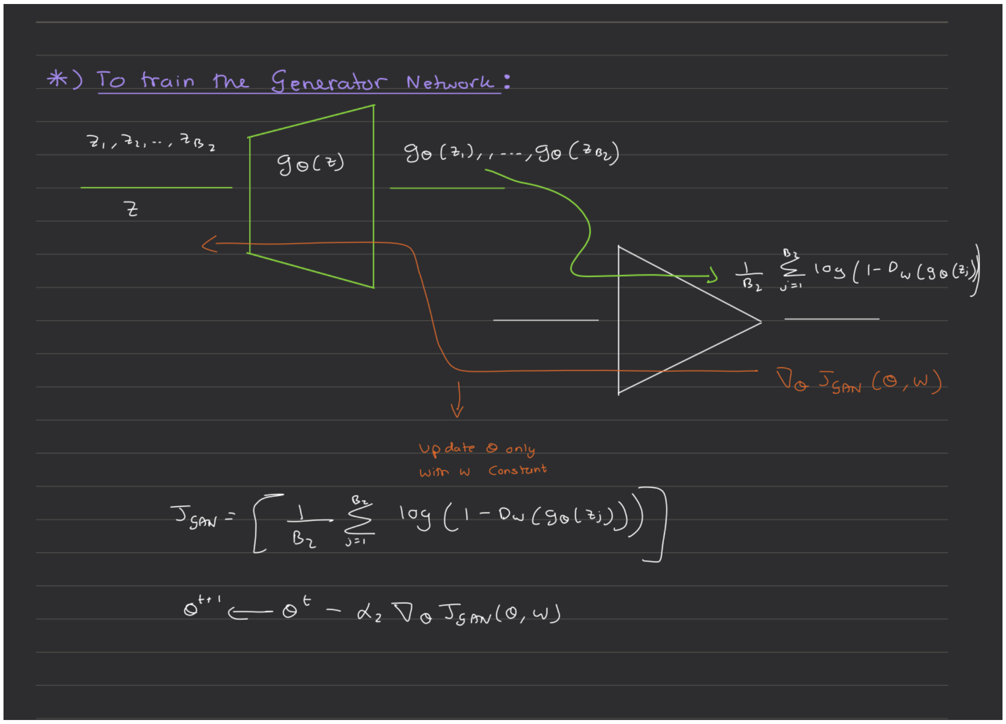


Figure 3: Train Generator network from notes

Once the discriminator is updated and fixed, we update the generator parameters θ to improve its ability to fool the discriminator.

We sample noise variables $z_1, z_2, \ldots, z_{B_2} \sim \mathcal{N}(0, I)$ and pass them through the generator:

$$\tilde{x}_j = g_\theta(z_j), \quad j = 1, \ldots, B_2$$

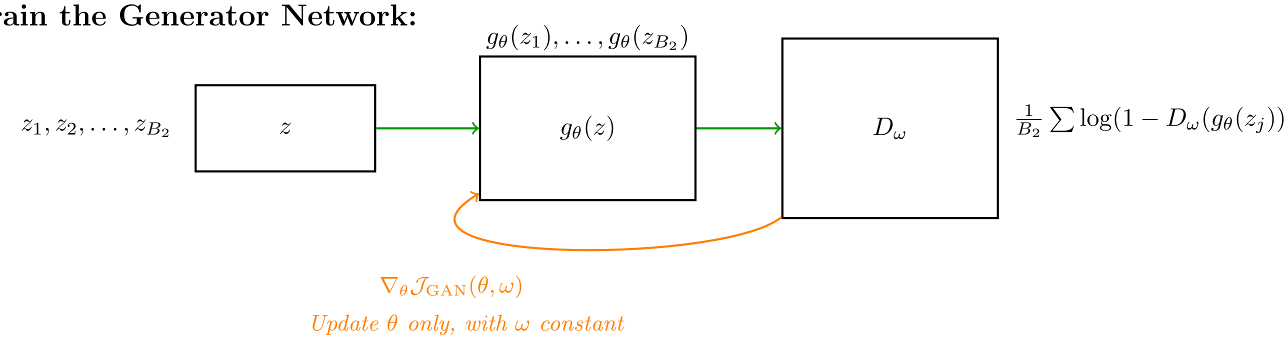
These generated samples are evaluated using the discriminator D_ω to compute the generator loss:

$$\mathcal{J}_{\text{GAN}} = \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_\omega(g_\theta(z_j)))$$

This loss encourages the generator to produce samples that are classified as real by the discriminator. The generator update step (keeping ω fixed):

$$\theta^{t+1} = \theta^t - \alpha_2 \nabla_\theta \mathcal{J}_{\text{GAN}}(\theta, \omega)$$

To train the Generator Network:



Why this works:

- The generator receives gradient feedback through the discriminator.
- The loss is minimized when $D_\omega(g_\theta(z)) \to 1$, i.e., the discriminator classifies generated samples as real.
- Training encourages the generator to match the real data distribution p_X.

Note: In practice, sometimes we maximize $\log(D_\omega(g_\theta(z)))$ to provide stronger gradients during early training.

This concludes the two-player minimax game formulation of GANs where the generator tries to fool the discriminator.

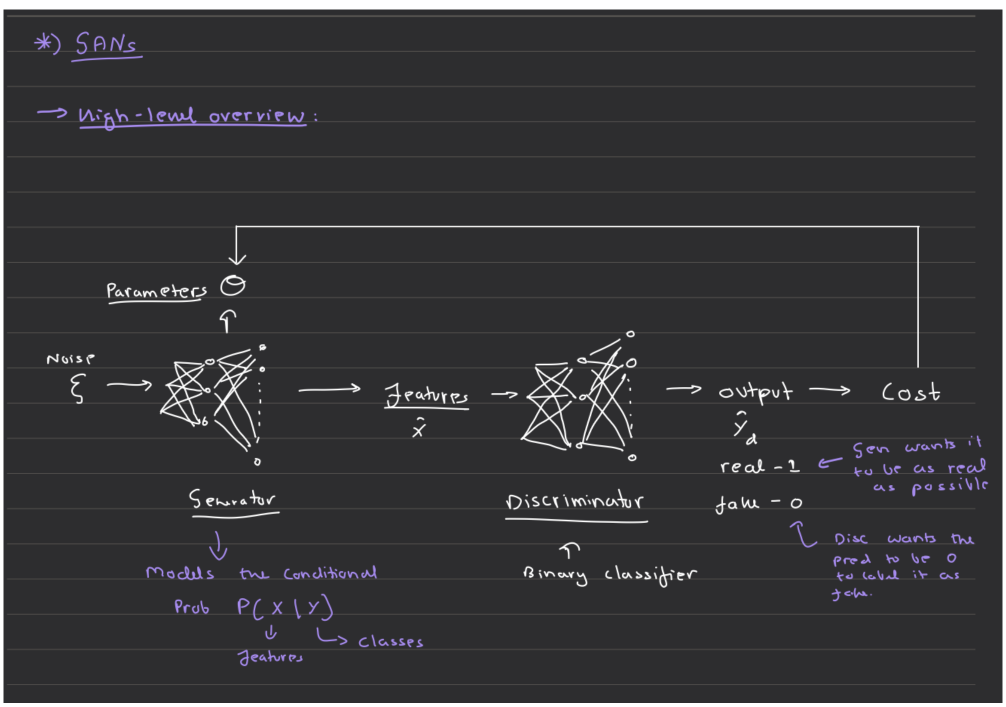


Figure 4: High-level overview from notes

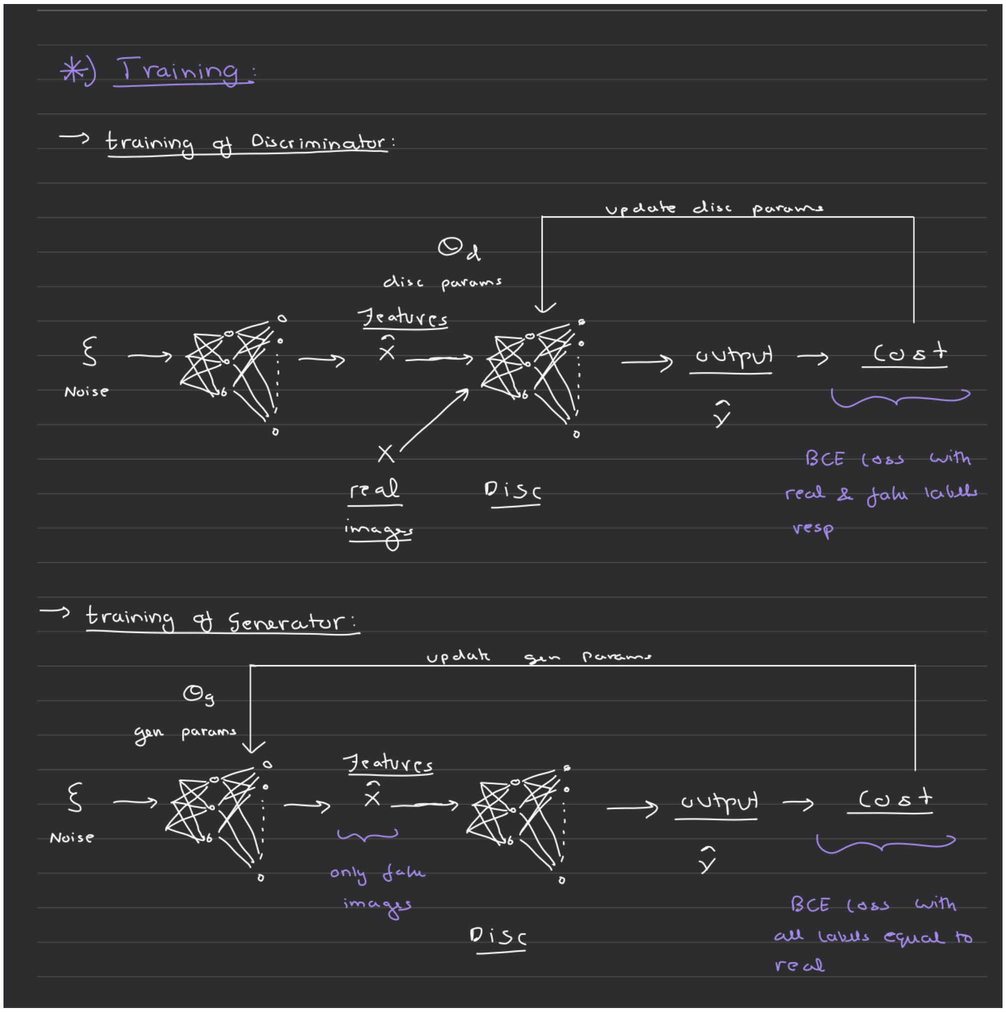


Figure 5: High-level overview from notes