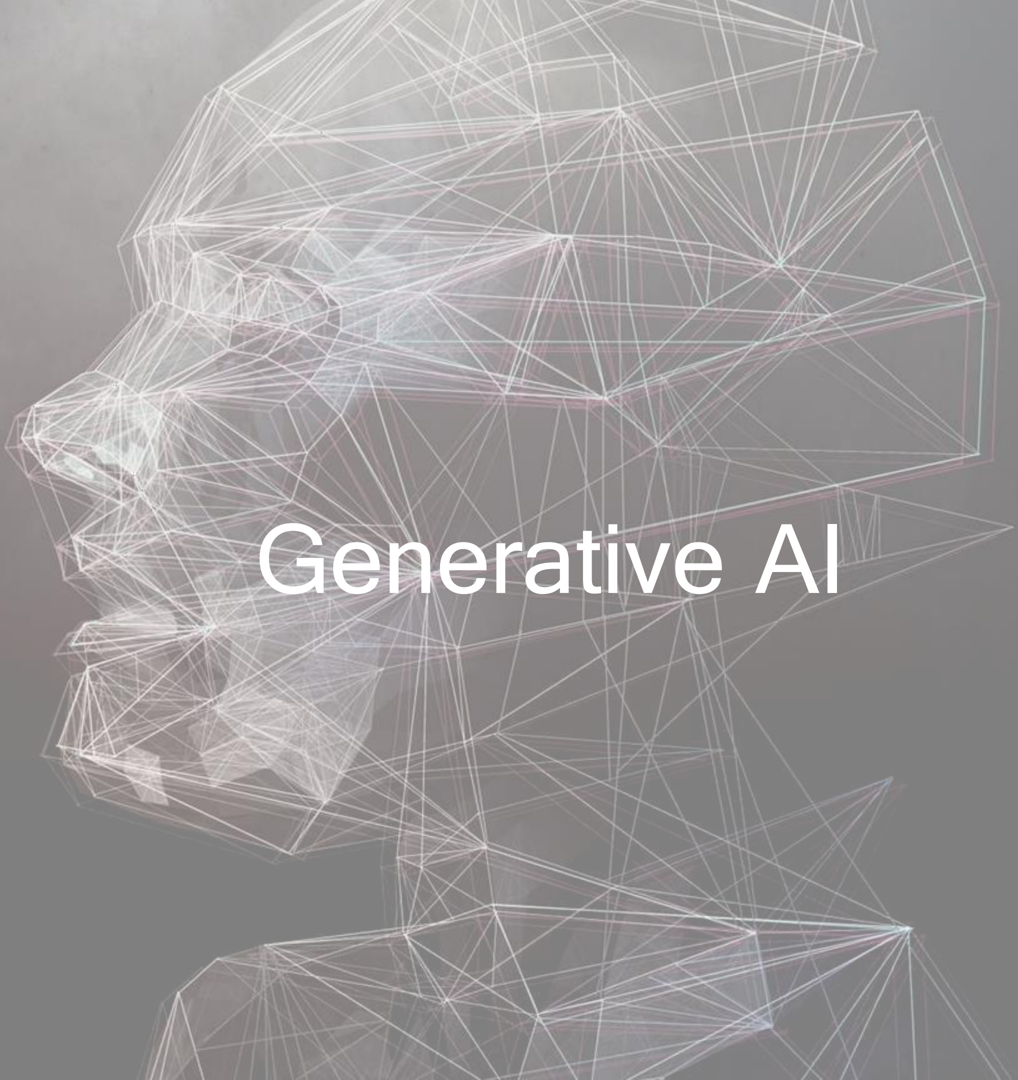
Generative AI

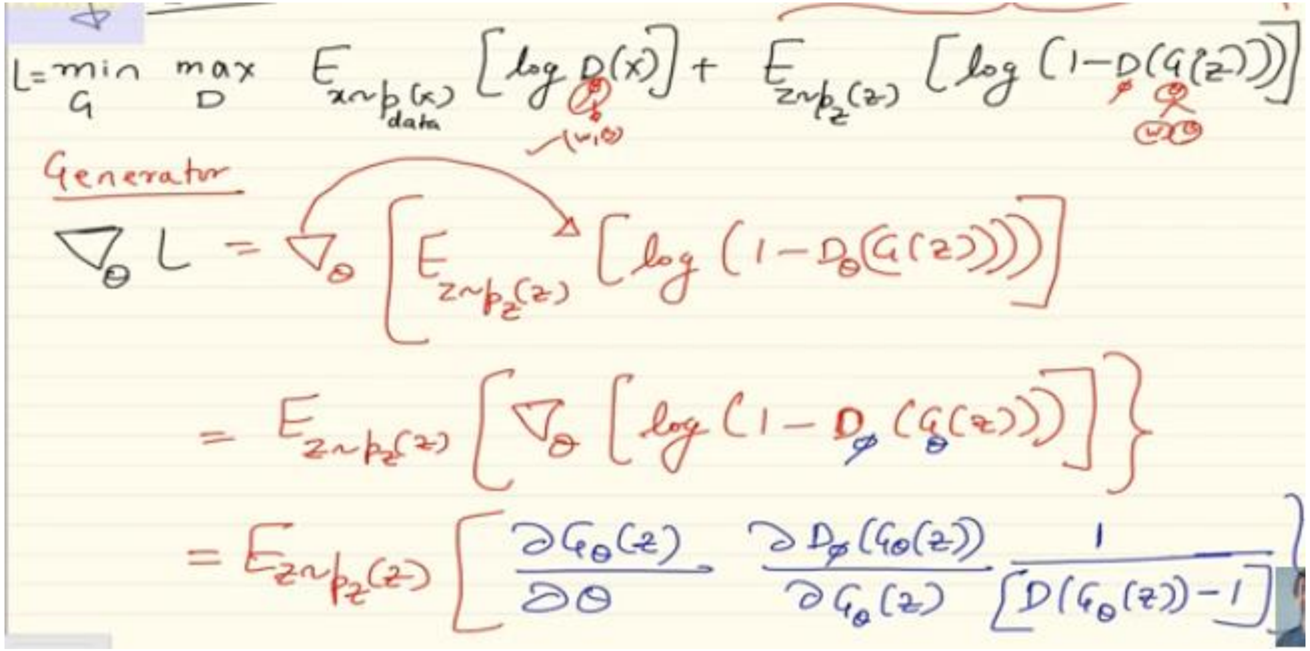


GAN Challenges

- Vanishing Gradient in Generators
- Oscillating loss
- Uninformative loss
- Mode collapse
- Hyperparameter Sensitivity

Vanishing Gradient issue

- Discriminator is trained first
- Gen is lousy in the beginning
- Backprop for gen
- Disc has already been trained



$$L = \min_{G} \max_{D} \; E_{x\sim p_{data}(x)}\left[\log \underset{\substack{\phi \\ (w,\theta)}}{D}(x)\right] + E_{z\sim p_z(z)}\left[\log\left(1-\underset{\phi}{D}(\underset{\substack{\theta \\ (w,\theta)}}{G}(z))\right)\right]$$

Generator

$$\nabla_\theta L = \nabla_\theta \left[E_{z\sim p_z(z)}\left[\log\left(1-D_\theta(G(z))\right)\right]\right]$$

$$= E_{z\sim p_z(z)}\left[\nabla_\theta\left[\log\left(1-D_\phi(G_\theta(z))\right)\right]\right]$$

$$= E_{z\sim p_z(z)}\left[\frac{\partial G_\theta(z)}{\partial \theta} \cdot \frac{\partial D_\phi(G_\theta(z))}{\partial G_\theta(z)} \cdot \frac{1}{[D(G_\theta(z))-1]}\right]$$

https://www.youtube.com/watch?v=ydGC0IvlCNs

$$\mathbb{E}_{x\sim p_g(x)}\left\{\frac{1}{(D(x)-1)}\frac{\partial D(x)}{\partial x}\frac{\partial x}{\partial \theta}\cdot\right]$$

$$\left\{\mathbb{E}_{x\sim p_g(x)}\left\{\frac{1\;\nabla_x D(x)\;\nabla_\theta x}{(D(x)-1)}\right\}\right. \qquad \nabla_x = \frac{\partial}{\partial x} \qquad \nabla_\theta = \frac{\partial}{\partial \theta}$$

Train Discriminator <
Train Generator

$$\tilde{D}\begin{cases} 1 & x\sim p_{data}(x) \\ 0 & x\sim p_g(x)\end{cases}$$

$D\to\tilde{D}$, $\tilde{x}\sim\tilde{p}_g$

$$\lim_{D\to\tilde{D}} D(x) = 0$$

$$\boxed{\lim_{D\to\tilde{D}}\nabla_x D(x) = 0}$$

$$\lim_{D\to\tilde{D}}\mathbb{E}_{x\sim p_g(x)}\left\{\frac{\overset{0}{\nabla_x D(x)}\,\nabla_\theta x}{D(x)-1}\right\} = 0$$

During the initial training disc might be too strong and gen is unable to generate good samples, hence disc always classify it as 0.
Thus, the gradient of D(G(z)) is always 0 or very close to 0.

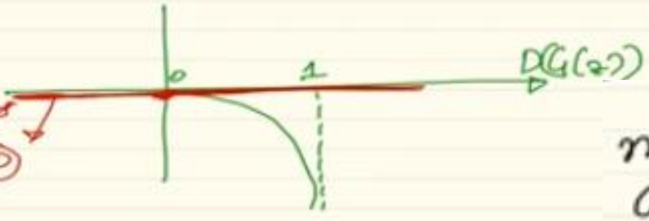
$$\min_{G} E_{z \sim p_z(z)} \left[\underline{\log\left(1 - D_{\phi}(G_{\theta}(z))\right)} \right]$$

$\textcircled{D}(G(z)) = 0$

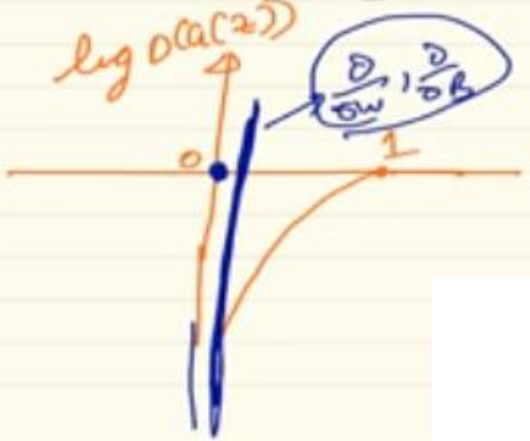
easy

(W, B)

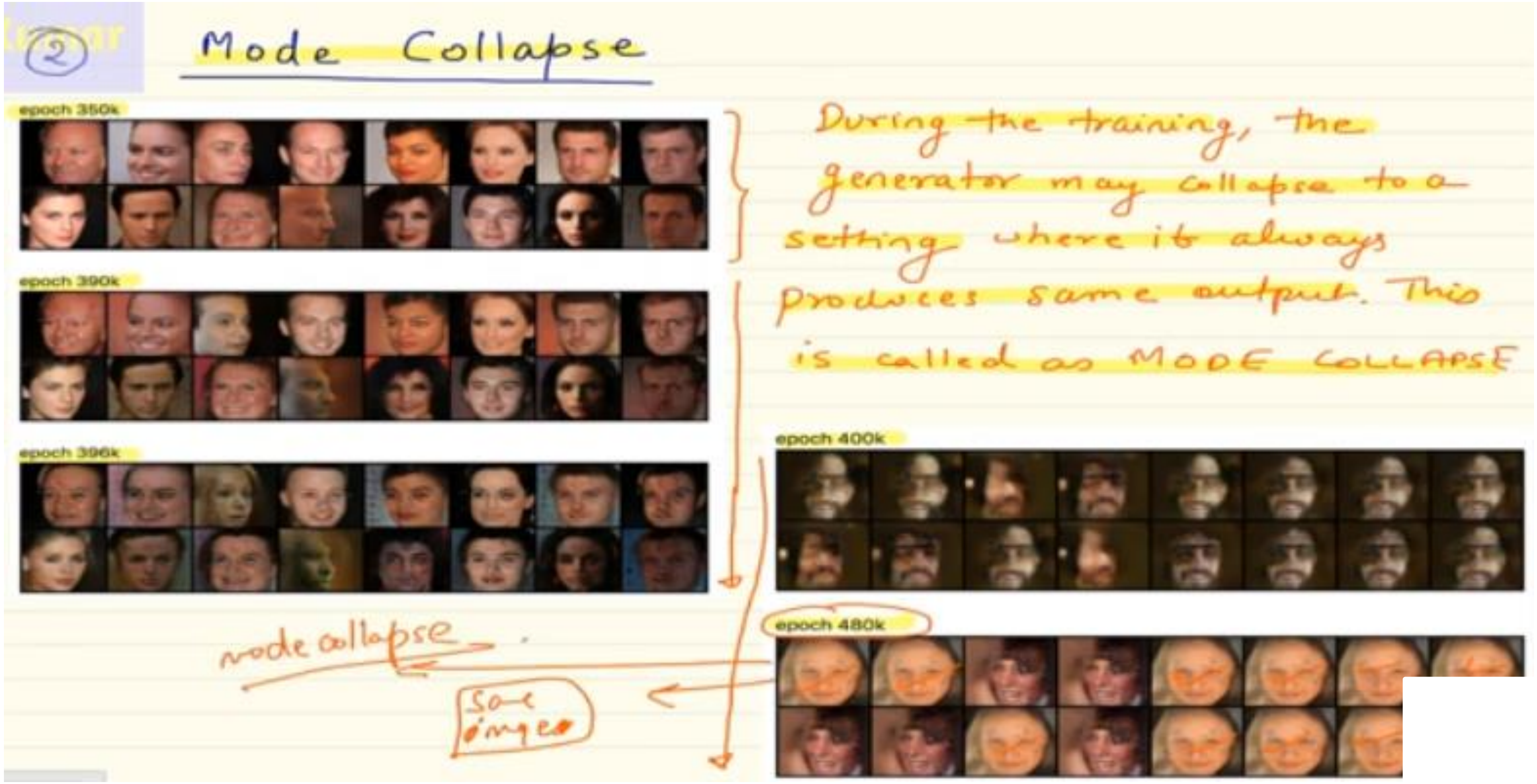
$R \rightarrow \textcircled{0}$



$$\max_{G} E_{z \sim p_z(z)} \log\left(\underbrace{D(G(z))}_{\textcircled{0}}\right)$$

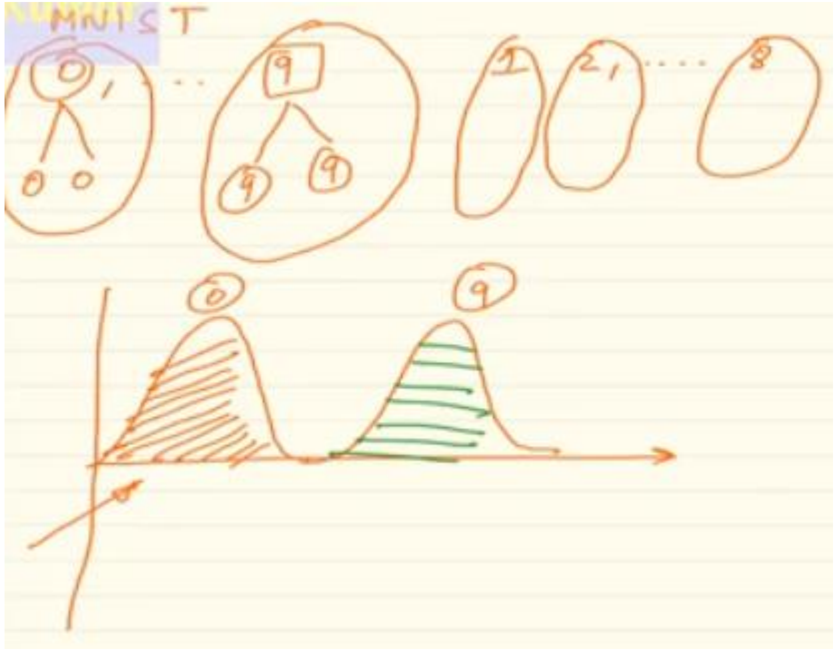


Mode Collapse



During the training, the generator may collapse to a setting where it always produces same output. This is called as MODE COLLAPSE

Mode collapse happens when the **generator fails to capture the full diversity of the real data distribution** and instead produces **only a few types (or even one type) of samples** repeatedly.



Dataset may be a multi-model distribution
All the 1's concentrated on one mode and may be 9s on the other mode.
GAN should be able to learn this multi-modal distribution, but it might learn only one mode distribution and hence produces only one kind of samples.

Gen just want to fool the discriminator, there is no constraint of diversity, so when Gen gets to know (G(D(Z))) where the disc is being fooled, Gen will sample from the known mode where it can fool the disc

If a GAN learns only one or two digits instead of generating all 10, it means **mode collapse has occurred**—it got stuck in learning only part of the data.

Oscillating Loss

- **Oscillating loss** refers to the situation where the loss functions of the generator and discriminator fluctuate erratically during training, instead of converging smoothly to an optimal solution.
- The losses oscillate (i.e., go up and down), which makes it difficult to monitor the training process.

- **Why it's a problem**:
- If the losses are oscillating, it often means that the GAN is not stabilizing its learning process.
- This can be a sign that the discriminator and generator are not reaching a good equilibrium, resulting in inefficient training and slow convergence.

Oscillating Loss

•**Potential solutions**:
•**Adjust the learning rates**: A too-high learning rate for either the generator or the discriminator can cause the model to oscillate. Lowering the learning rate or using learning rate schedules can help.

•**Use of batch normalization**: Normalizing the intermediate layers of the generator and discriminator helps stabilize training by ensuring the activations stay within a certain range.

•**Two-time scale update rule (TTUR)**: This technique involves using *different learning rates* for the generator and discriminator, typically with a lower learning rate for the generator to ensure it doesn't overpower the discriminator.

Hyperparameter Sensitivity

•GANs are highly sensitive to hyperparameters, including learning rates, batch size, the number of layers, and the choice of activation functions.

•Even small changes in these hyperparameters can lead to drastically different results, making the training process difficult to control and tune.

•**Why it's a problem**:Since GANs can be sensitive to a wide range of hyperparameters, finding the right configuration for stable and effective training is often time-consuming and trial-and-error-based. This makes deploying GANs in real-world applications challenging

Evaluating GANs Output

For Tabular Datasets

- **Machine Learning Efficacy**
  - Train a classifier on **generated data** and test it on **real data**.
  - If the classifier performs well, the generated data captures real data distributions.

Example: If a GAN generates synthetic patient records, we train a heart disease classifier on synthetic records and test it on real patients. If accuracy is close to real-data-trained models, the GAN-generated data is useful.

- **Feature Distribution Similarity**
  - Compares statistical properties (mean, variance, correlation) between real and synthetic data.

Example: If a GAN generates synthetic bank transactions, we check if distributions of amounts, transaction times, and categories match real transaction distributions.

For Images:

- **Inception Score (IS)** - Measures Image Quality & Diversity
- It relies on a pre-trained **Inception model** to classify the generated images and uses the concept of **KL divergence** to measure two key factors: **realism** and **diversity**.

- **Classifying Images**: For each generated image, the Inception model predicts the probability distribution over multiple classes (like "dog", "cat", "car", etc.). P(y|x)

- **Diversity of Classes**: We want to check if the GAN generates images from **different categories**. If all images are about the same thing (e.g., all dogs), there's low diversity. Calculate the **average distribution** of predicted classes across the entire dataset of generated images. P(y)

KL divergence: If **p(y|x) is very different from p(y)**, the KL divergence is **high and Low IS**

If KL is high, it means low quality, model is less confident and may be no diversification.

1️⃣ **Classify Each Image:** Inception model gives class probabilities $p(y|x)$.

2️⃣ **Compute $p(y)$:** Average class distribution across all generated images.

3️⃣ **Compute KL Divergence:** Measure how different $p(y|x)$ is from $p(y)$.

4️⃣ **Compute IS:** Average KL divergence across all images and exponentiate it.

✅ High IS → Realistic & diverse images.

❌ Low IS → Unclear images or mode collapse.

$$D_{\mathrm{KL}}(p(y|x) \parallel p(y)) = \sum_y p(y|x) \log \frac{p(y|x)}{p(y)}$$

•**Sharp p(y|x)** (high confidence for each image) does **not necessarily lead to a higher IS** unless it is consistent with p(y). High KL divergence means lower IS, because it indicates poor alignment between per-image class predictions and the overall generated distribution.

•If the generated images are diverse, the marginal distribution p(y) will be close to uniform (i.e., all classes are equally represented).

P(y|x) is Low (Uncertain Prediction per Image) but P(y) is High (High Diversity) KL Divergence would be **high? Yes, hence low IS**

•**IS can be fooled by mode collapse? True?**

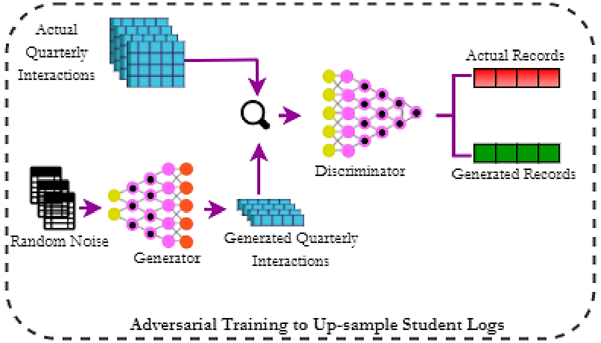
Fréchet inception distance(FID)

- FID not just passes the generated images, but also the real images i.e. ground truth through the inceptionv3 network.
- **Extract features** from both real & generated images.
- **Compute mean & covariance** of feature distributions.
- **Compare the two distributions** (generated vs actual) using Fréchet Distance.
- **Lower FID = better quality & realism**

If the means and covariances are close, FID is low → GAN is generating high-quality images.
If they are far apart, FID is high → GAN is producing unrealistic or low-diversity images.

Conditional Generative Adversarial Networks

- The y in $D(x|y)$ or $G(z|y)$ is the **conditional information** (e.g., class label), which we use to guide the generator and discriminator.
- The **label** (0 or 1) used in the objective function refers to whether a sample is real (label 1) or fake (label 0) and is used to train the discriminator and generator based on this classification.


Actual Quarterly Interactions
Actual Records
Discriminator
Generated Records
Random Noise
Generator
Generated Quarterly Interactions
Adversarial Training to Up-sample Student Logs

$$\min_{G}\max_{D} f(D,G) = E_D + E_G$$

$$\boldsymbol{E_D = E_{x\sim p_{data}(X)}\underbrace{[\log D(x|y))]}_{\text{x is the actual data distribution, label 1}} + E_{z\sim p_z(z)}[\log(1 - \underbrace{D(G(z|y)))]}_{\text{generated data, passed to disc with label 0}}}$$

$$E_G = E_{z\sim p_z(Z)}[\log(1 - D(G(z|y)))]$$

This term encourages the discriminator to correctly classify generated samples G(z|y)as fake, i.e., to assign a low probability to them being real.

Conditional Generative Adversarial Networks

$$\min_{G} \max_{D} f(D,G) = E_D + E_G$$

$$E_D = E_{x\sim p_{data}(X)}[\log D(x|y))] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]$$

$$\boldsymbol{E_G = E_{z\sim p_z(Z)}[\log(1 - \underline{D(G(z|y)})))]}$$

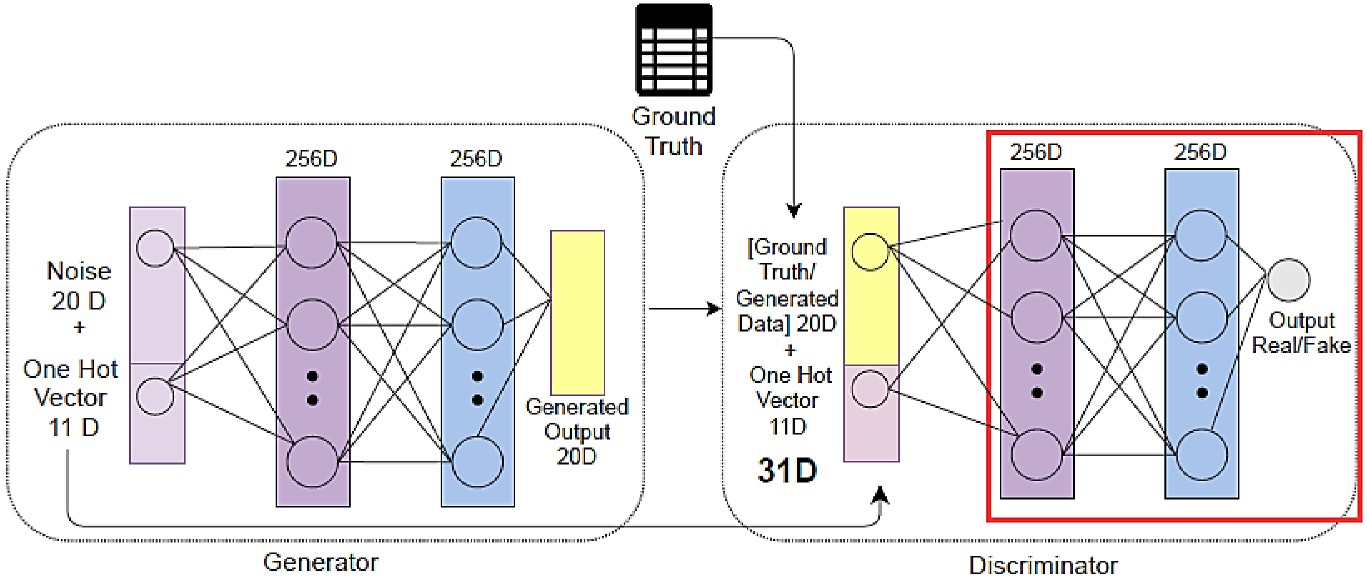
Generated data distribution, label 1

The generator wants D(G(z|y)) to be as high as possible, i.e., it wants the discriminator to classify its generated samples G(z|y) as real.
Minimizing it means that generator is trying to "fool" the discriminator into thinking that the generated samples are real.

Detailed CGAN Architecture

The discriminator also takes the condition y as input along with the sample (real or generated).

It learns to distinguish real samples x from generated samples $G(z|y)$, given the context y.



GANs vs VAEs

- **Pros and Cons of GANs**:
- **Pros**: High-quality sample generation, especially for images, and ability to model complex data distributions.
- **Cons**: Training instability, mode collapse, and difficulty in generating structured data (like text).

- **VAEs**:
- **Pros**: Easier to train, provide a continuous latent space (which is useful for interpolation), and are better suited for structured data.
- **Cons**: Generated samples often lack sharp details and can be blurry compared to GANs.