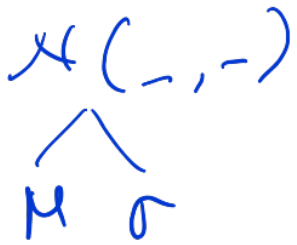
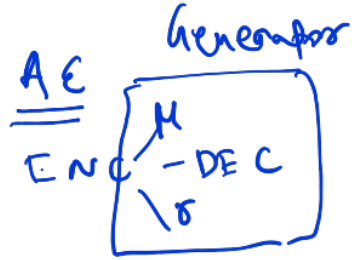
# Generative Adversarial Networks (GANs)

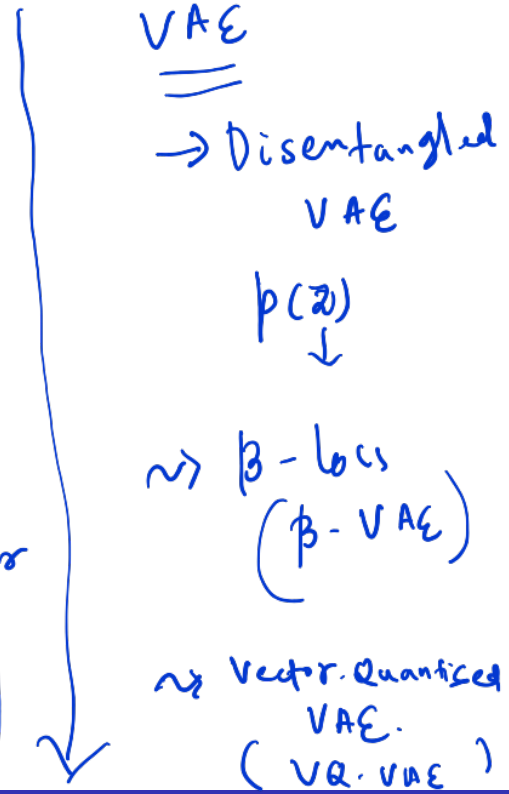
Tirtharaj Dash

Dept. of CS & IS and APPCAIR
BITS Pilani, Goa Campus

December 07, 2021







# Main Idea

- GAN consists of two different models:
  - **Discriminative model:** estimates the conditional prob. $p(y|\mathbf{x})$ of the label $y$, given the data instance $\mathbf{x}$. (e.g. MLP, CNN)
  - **Generative model:** estimates the joint prob. $p(\mathbf{x}, y)$
- Note: Discriminative model can only be used in an supervised setting; whereas geneative model can be used in both supervised as well as in unsupervised setting.
- Example: In VAE, a Gaussian distro is used to generate sample and passed to a decoder to (re-)construct the original data instance.

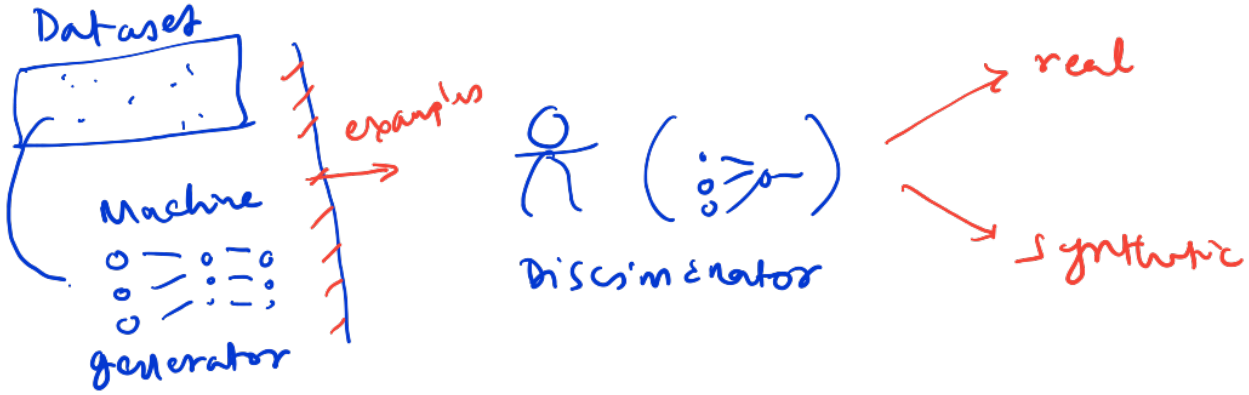
# Main Idea

GAN — Generator (DNN)
Discriminator (DNN)

- Generative model inside a GAN:
  - It produces synthetic examples of objects that are simple to a real repository of examples.
  - The goal is to create synthetic objects that are so realistic that it is hard for a trained observer to say that it is synthetic.

dataset

# Main Idea



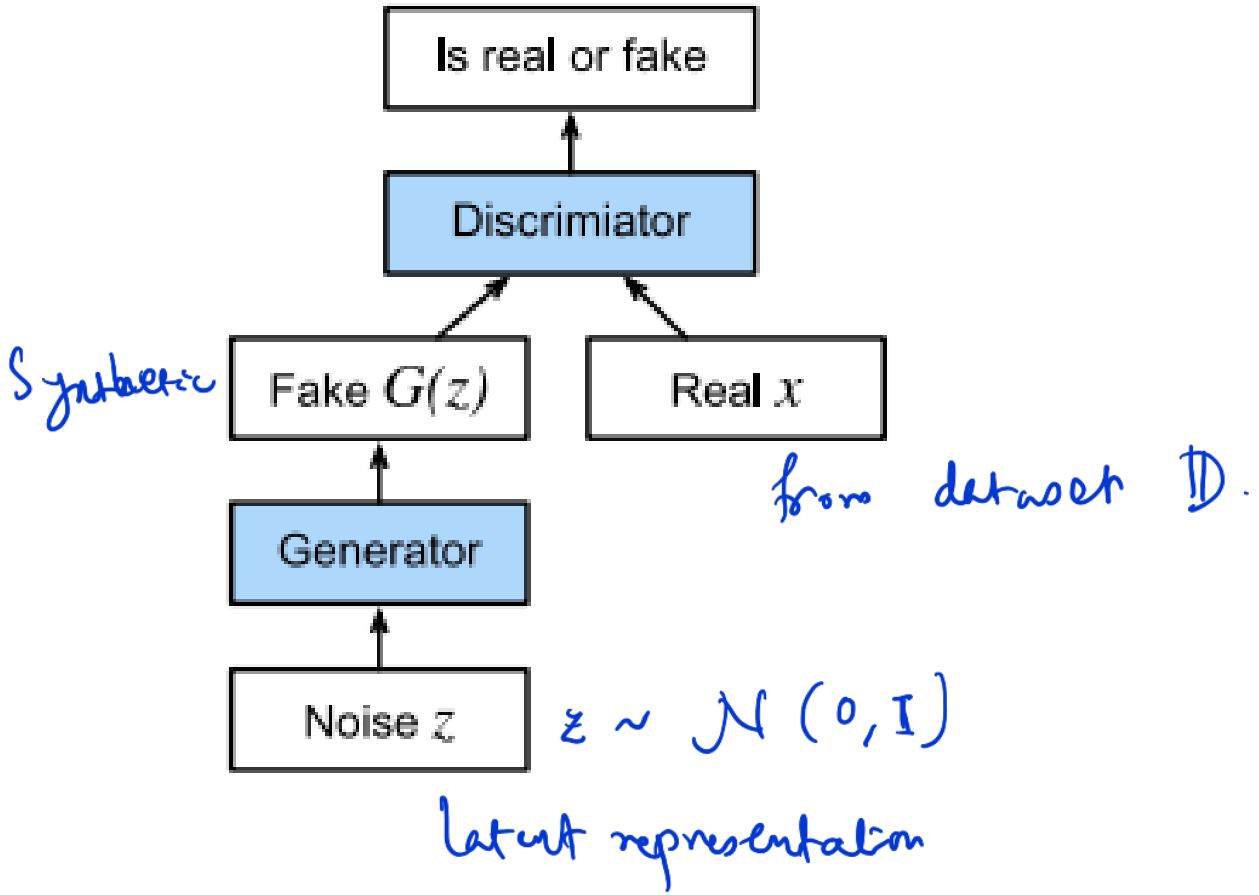

- Discriminative model inside GAN:
  - It takes in inputs of either real examples from the base data or synthetic objects created by generator and tries to discriminate between them.
  - The goal is to correctly classify the real and synthetic objects.

# Main Idea

- Why "adversarial"? The goals of the generator and the discriminator are quite opposite, i.e. the generator wants to fool the discriminator which does not want to be fooled.

# Main Idea

GAN architecture:



# Training a GAN

Discriminator:

real or synthetic (binary clf.)

- Let $\mathbf{x}$ be any data instance. The discriminator is a binary classifier to distinguish if $x$ is real (for real data) or synthetic (from generator).
- This can be written as:

Discriminator function $D(\underline{x}; \underline{w})$

$p(y|\underline{x})$ $\underline{x} \in \mathbb{D}$ $G(z)$

$$D(\mathbf{x}) = \frac{1}{1 + \exp(-o)}$$

sigmoid (σ)

for some output $o \in \mathbb{R}$ computed at the output neuron of the discriminator.

- Let the label $y$ be 1 for real data and 0 for synthetic data.
- The training minimises the cross-entropy i.e.

$$\min_D \{ \underbrace{-y \log D(\mathbf{x})}_{p(y|\underline{x})} - (1 - y) \log(1 - D(\mathbf{x})) \}$$

Images: D: CNN.

# Training a GAN

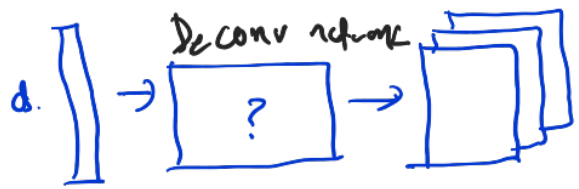
Latent representation $z \sim \mathcal{N}(0, I)$ (noise) $\longrightarrow$ synthetic data instance

Generator: $h \times w \times d$

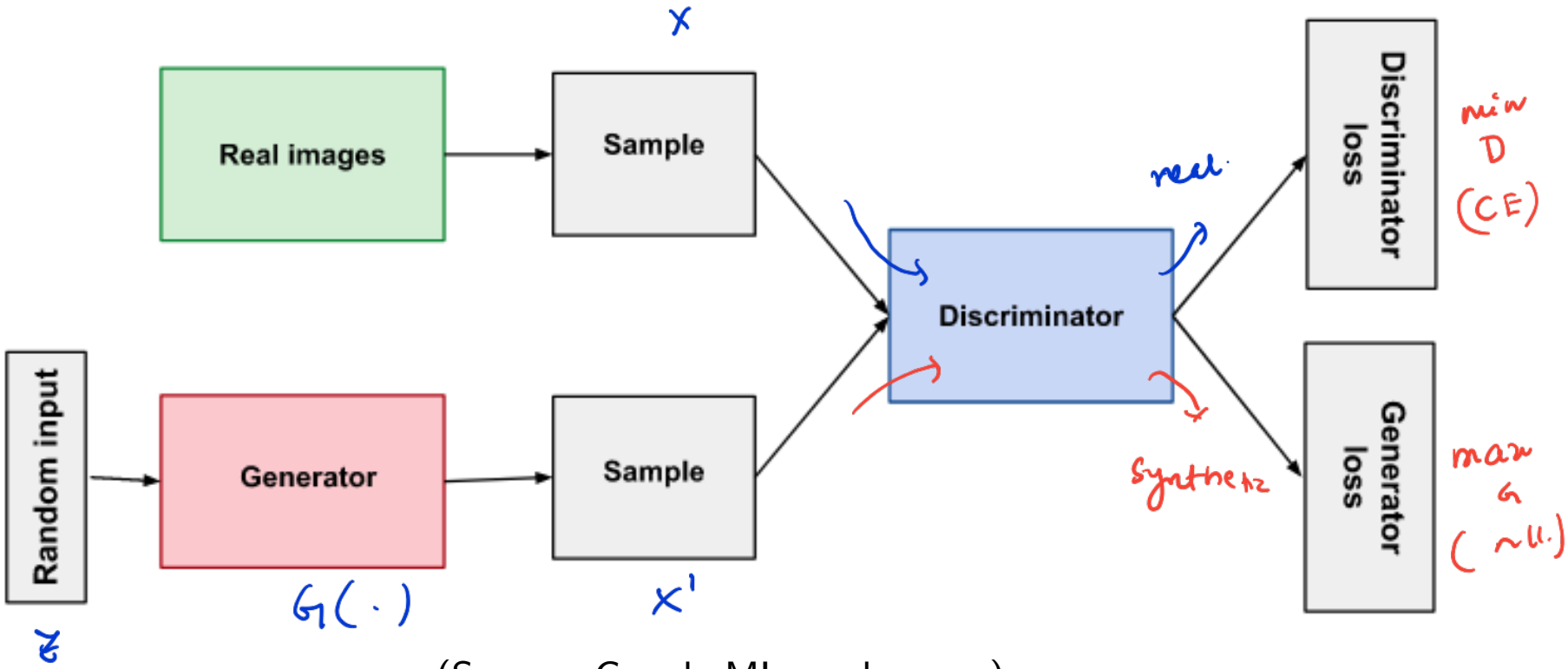
- It first draws some parameter $\mathbf{z} \in \mathbb{R}^d$ from some some distro. e.g. $\mathbf{z} \sim \mathcal{N}(0, 1)$.
- It then applies a function on $\mathbf{z}$ to generate an instance: $\mathbf{x}' = G(\mathbf{z})$ (like a decoder network)
- The goal of the generator is to produce $\mathbf{x}'$ such that $D(\mathbf{x}') \approx 1$.
- That is: For a given discriminator $D$, we update the parameters of the generator $G$ to maximise the cross-entropy loss when $y = 0$, i.e.

$$\max_G\{-(1-y)log(1-D(G(\mathbf{z})))\} = \max_G\{-log(1-D(G(\mathbf{z})))\}$$

$-(1-0)\log(1-D(x'))$ $\qquad$ $x'$



# Training a GAN




(Source: Google ML crashcourse)

# Training a GAN

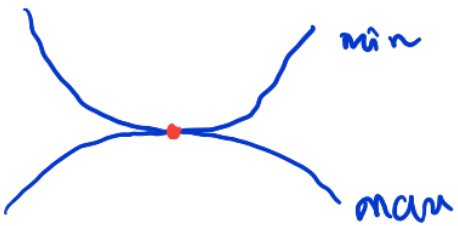
Summary:

- The discriminator and the generator are playing a "minimax" game with one comprehensive objective function

$$\min_{D} \max_{G} \{ -E_{\mathbf{x} \sim \mathcal{X}} \log D(\mathbf{x}) - E_{\mathbf{z} \sim \mathcal{N}(0,1)} \log(1 - D(G(\mathbf{z}))) \}$$

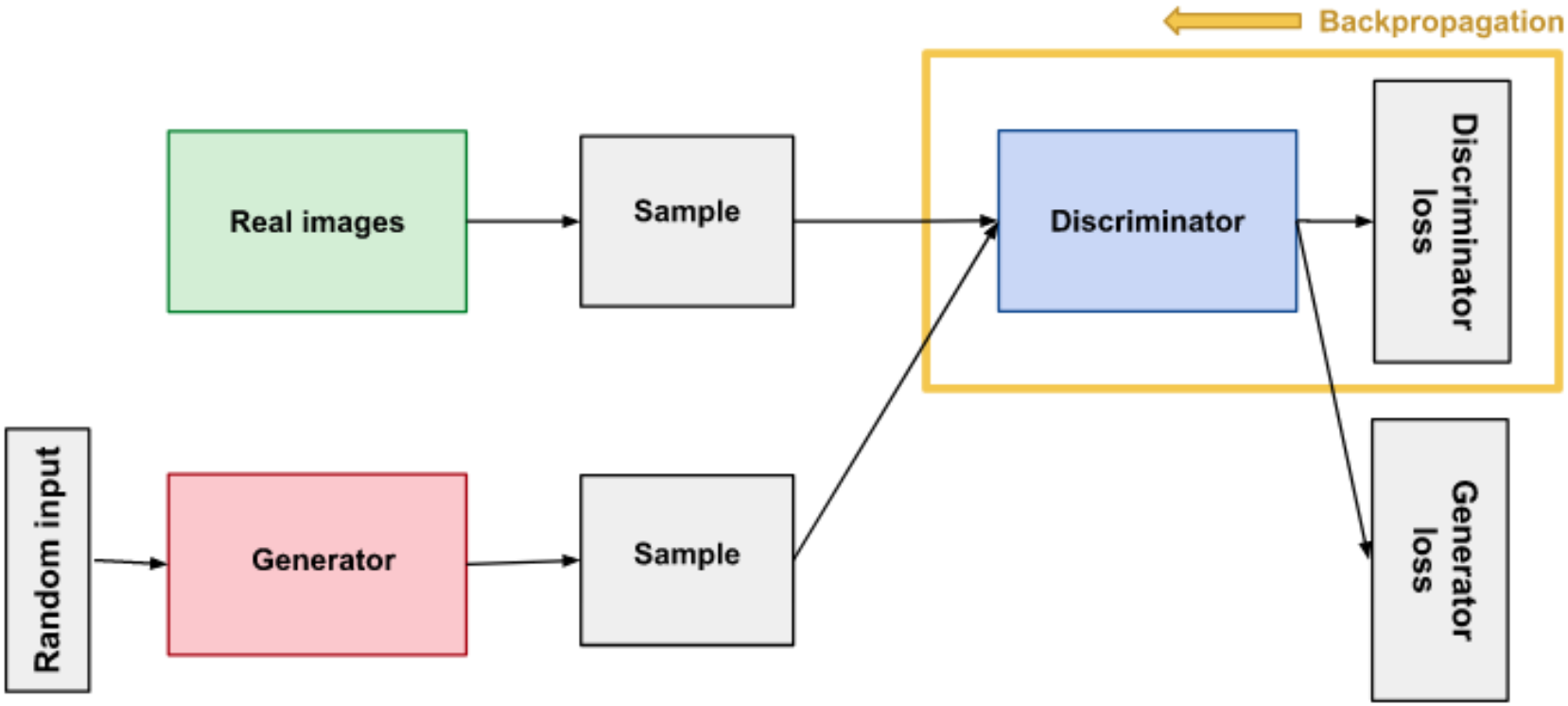
$y = 1$ (real data) $\quad$ noise $\quad$ $y = 0$ (generated data)

- The solution is a saddle point (Can you see this?)



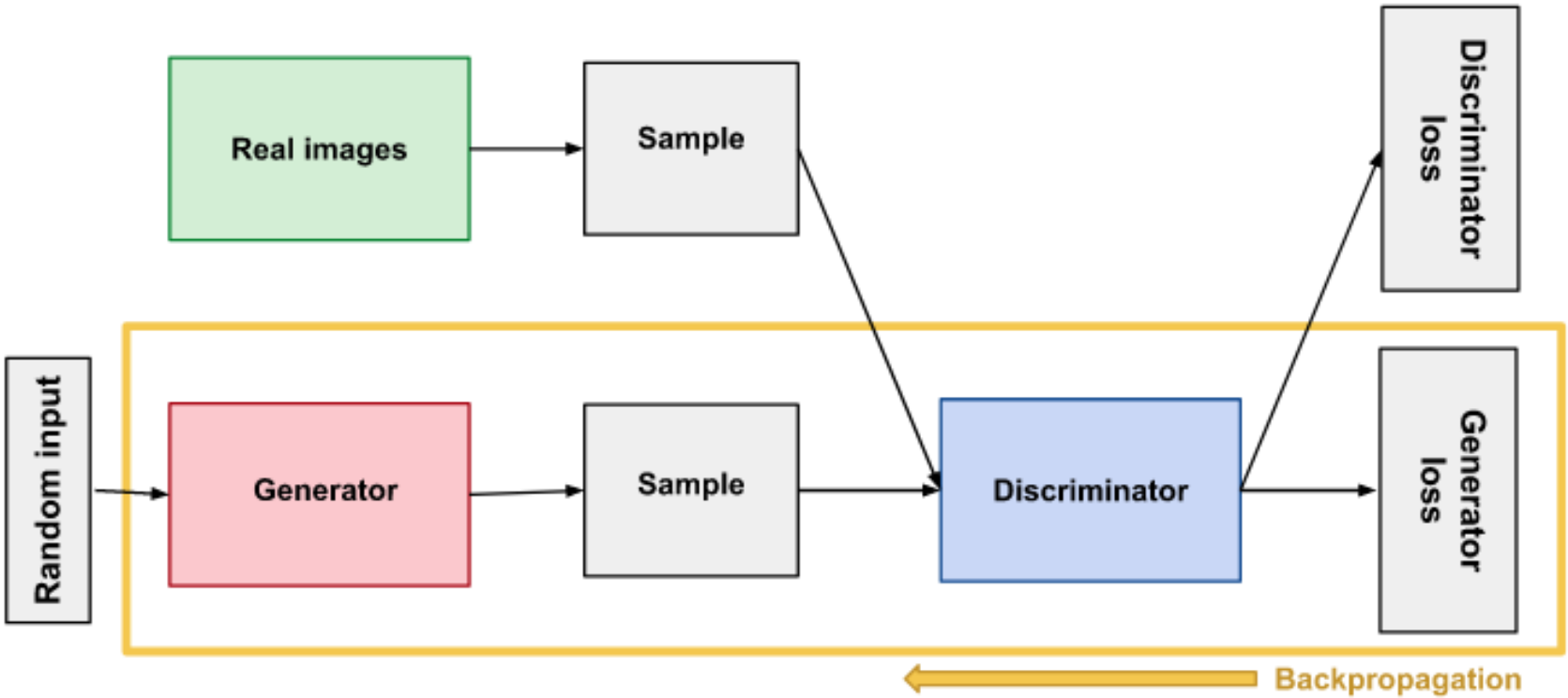
# Training a GAN

Backprop:




(Source: Google ML crashcourse)

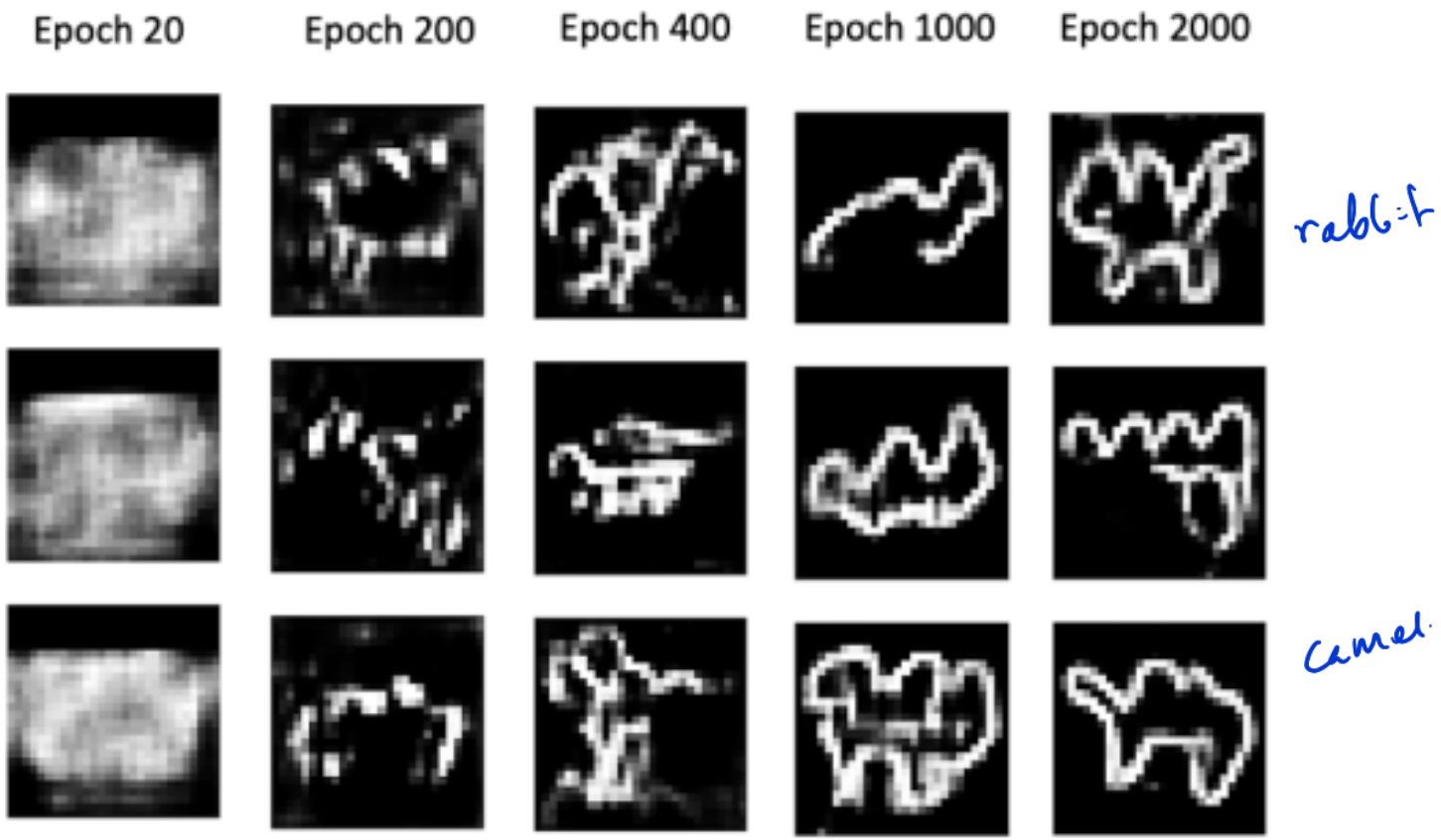
# Training a GAN




(Source: Google ML crashcourse)

# Training a GAN

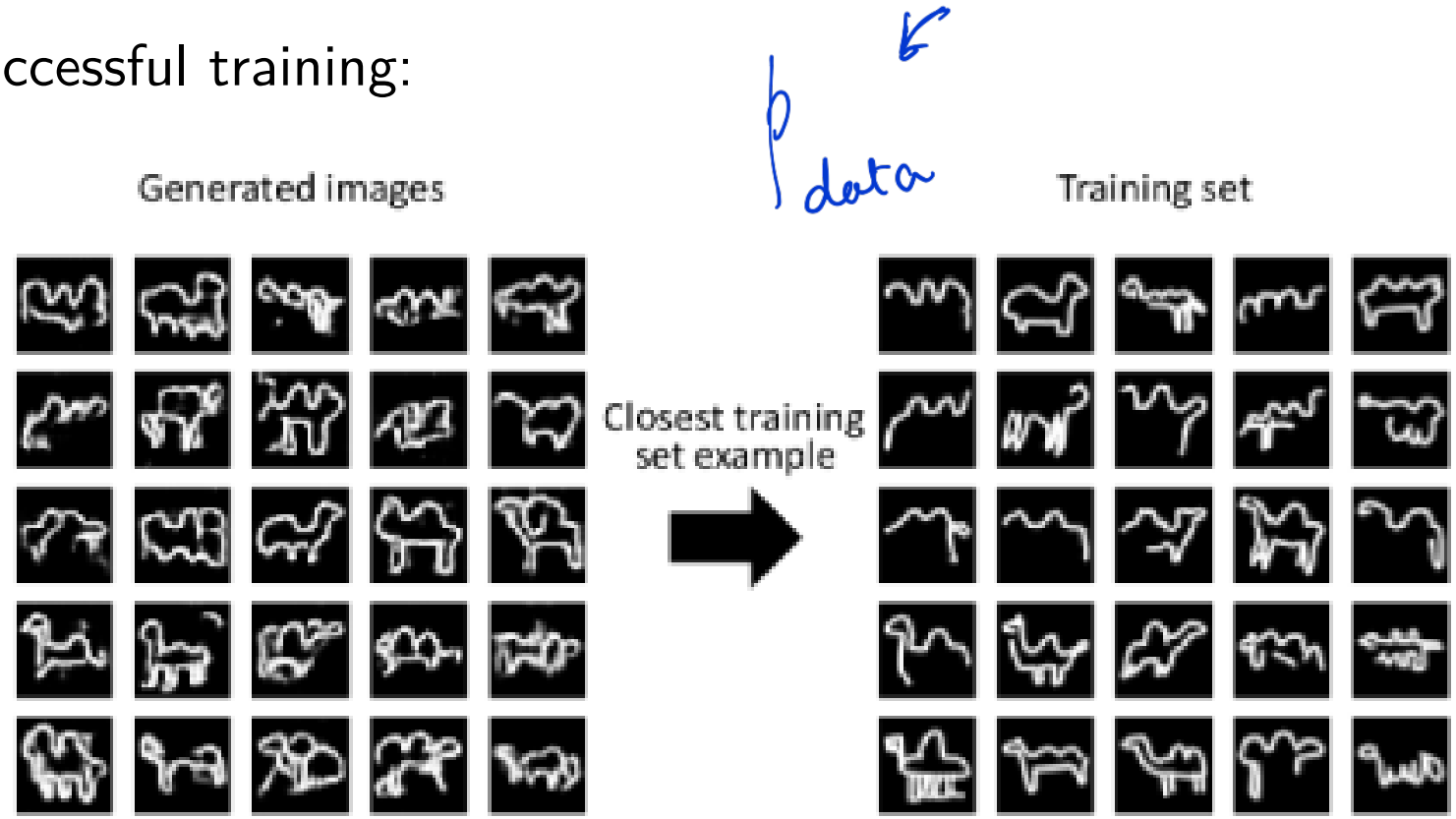
Generated samples at some specific epochs:




(Source: David Foster, Generative Deep Learning)

# Training a GAN

With successful training:

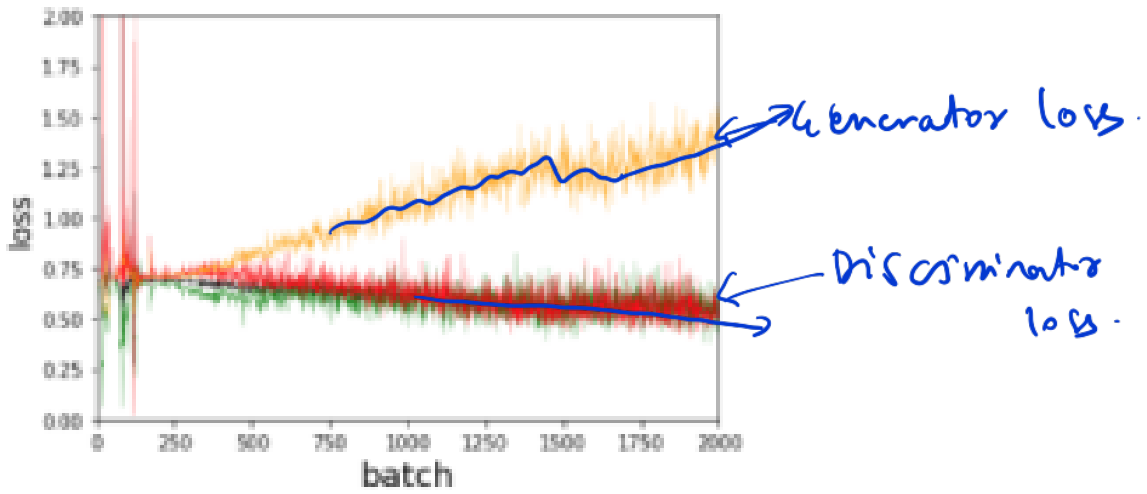



(Source: David Foster, Generative Deep Learning)

# Known challenges in GAN Training

Oscillating Loss:

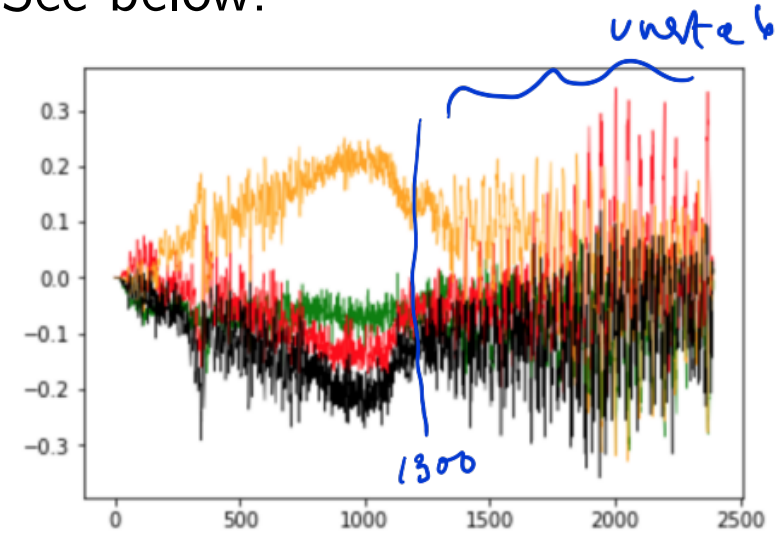
- The loss of the discriminator and generator can start to oscillate wildly, rather than exhibiting long-term stability.
- Typically, there is some small oscillation of the loss between batches. However, it the long term the loss should be *stable* (or gradually decreases or increases). See below:




(Source: David Foster, Generative Deep Learning)

# Known challenges in GAN Training

- But, most of the time, the loss of the discriminator and generator start to spiral out. See below:




(Source: David Foster, Generative Deep Learning)

- Also, it is difficult to establish if or when this might occur during training.
- This issue is very common in vanilla GANs.

# Known challenges in GAN Training

Mode Collapse:

local minimum

- Mode collapse occurs when the generator finds a small number of samples that fool the discriminator.
- It isn't able to produce any examples other than this limited set.

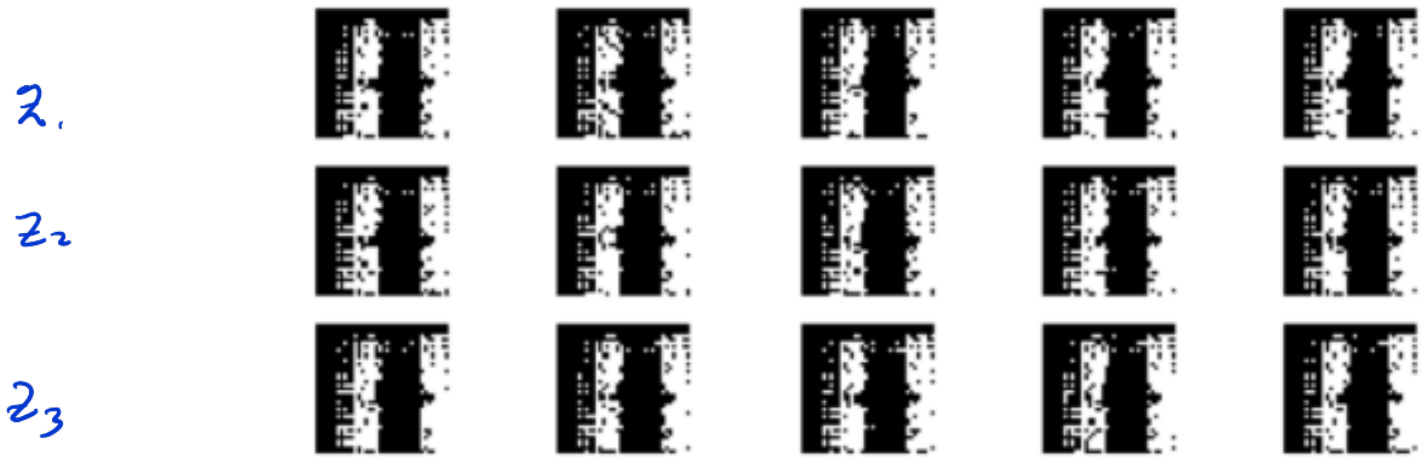
# Known challenges in GAN Training

- How does this occur?
  - Suppose we train the generator over several batches without updating the discriminator in between.
  - The generator would be inclined to find a single observation (also known as a mode) that always fools the discriminator and would start to map every point in the latent input space ($\mathbf{z}$) to this observation.
  - This means that the gradient of the loss function collapses to near 0.

$z \longrightarrow$ mode.

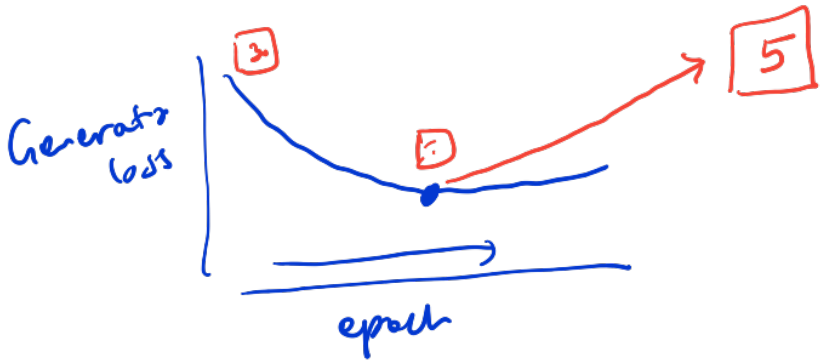
# Known challenges in GAN Training

- Can retraining the discriminator help?
  - It maynot: The generator will simply find another mode that fools the discriminator.
  - That is, the generator has become numb to the inputs and therefore does not diversify the generation (see below: generator generates identical samples even though its inputs are different).




(Source: David Foster, Generative Deep Learning)

# Known challenges in GAN Training

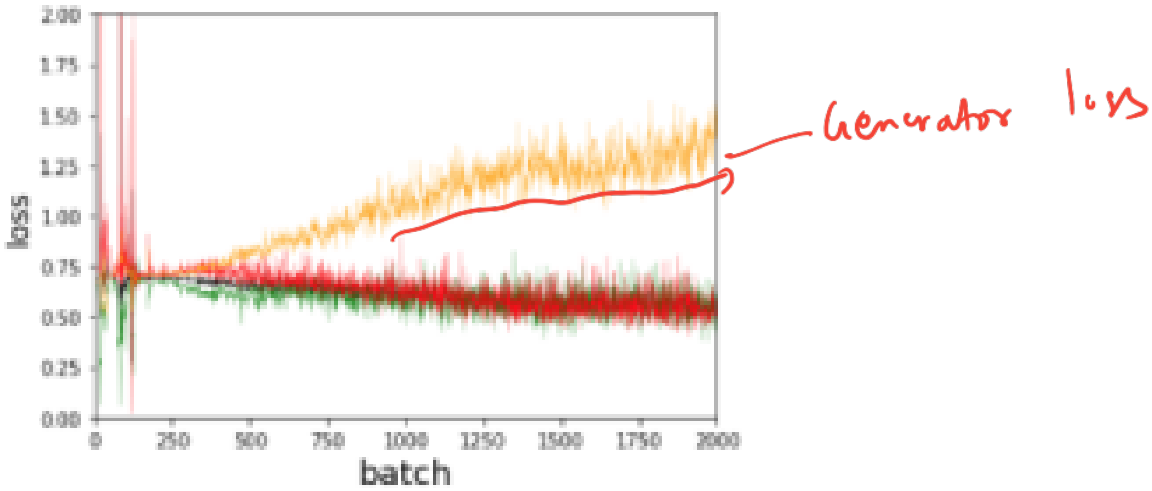



Uninformative Loss:

- It is natural to think that smaller the loss function of the generator, the better the quality of the images produced.
- However, since the generator is only graded against the current discriminator and the discriminator is constantly improving, we cannot com- pare the loss function evaluated at different points in the training process.

# Known challenges in GAN Training

- See below: The loss function of the generator actually increases over time, even though the quality of the images is clearly improving.




(Source: David Foster, Generative Deep Learning)

- This lack of correlation between the generator loss and image quality sometimes makes GAN training diffi- cult to monitor.

# Known challenges in GAN Training

Hyperparameter sensitivity:

- GANs involve a huge set of hyperparameters: generator and discriminator structure, training hyperparameters, etc.
- GANs are highly sensitive to very slight changes in all of these parameters, and finding a set of parameters that works is often a case of educated trial and error, rather than following an established set of guidelines.

# Known challenges in GAN Training

Some of the above challenges (mainly, mode collapse) can be handled with Wasserstein GAN: https://arxiv.org/abs/1701.07875

Wasserstein GAN - Gradient Penalty.

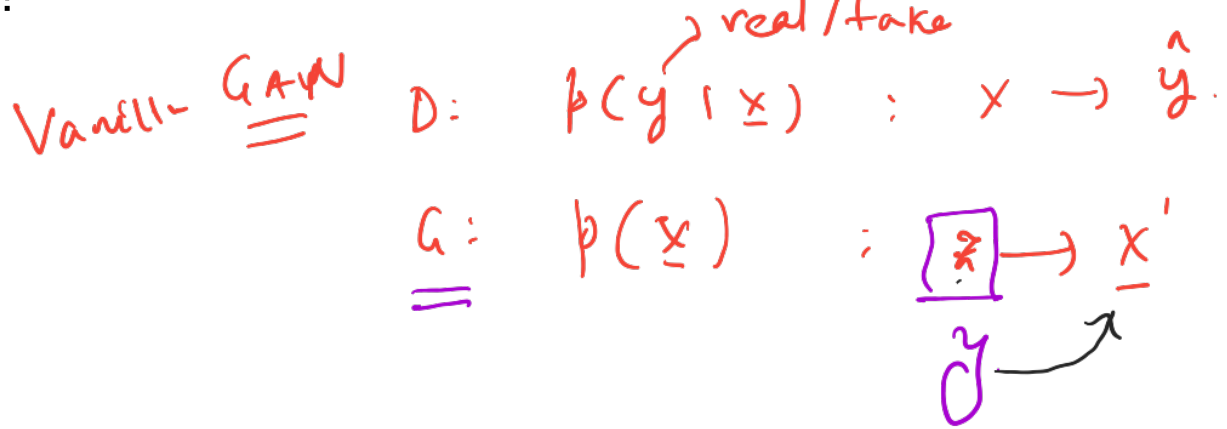
GAN Generate digits. (dataset: MNIST)

Q. Generate 10 images of the digit "7".

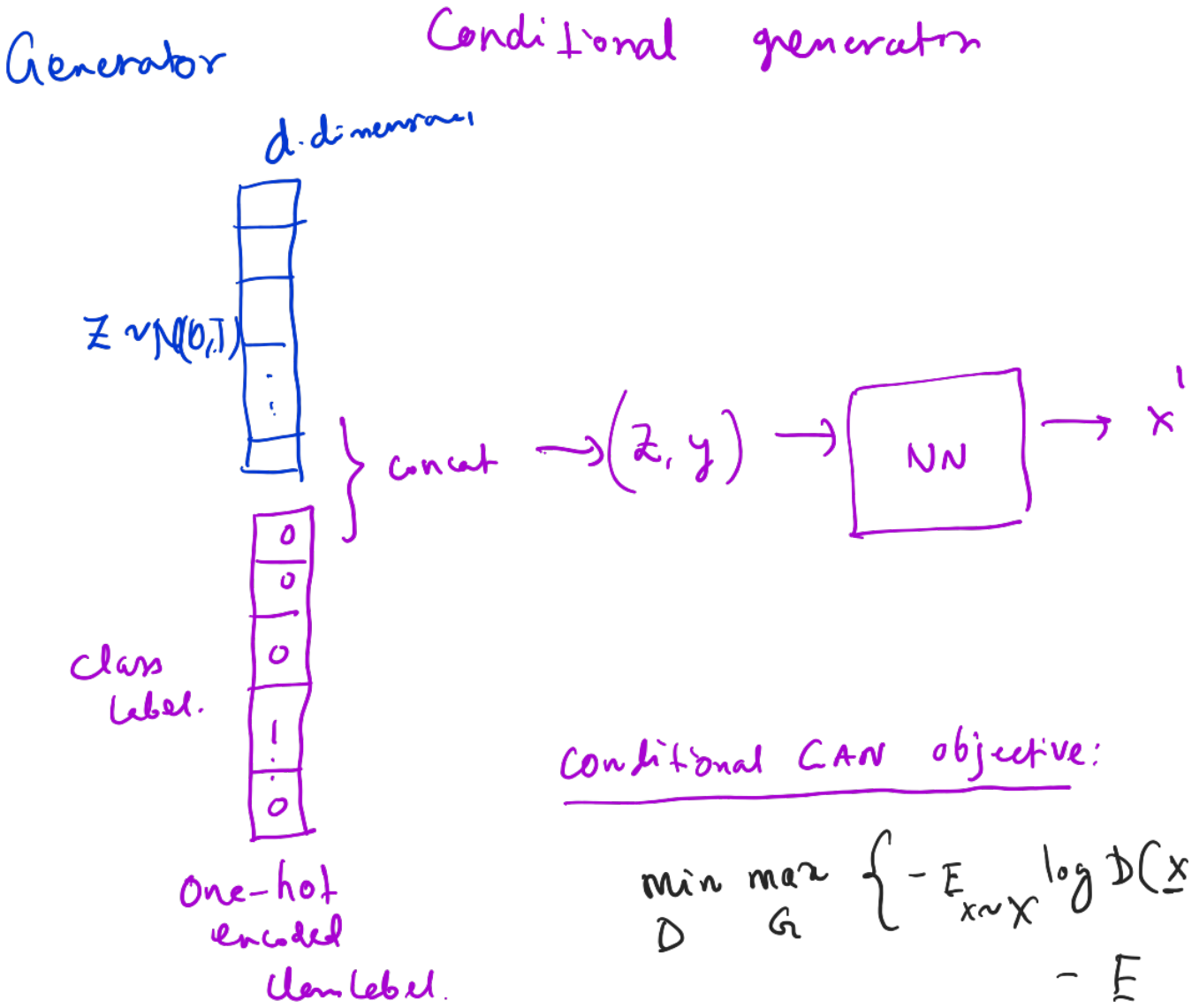
"3"

# Conditional Generation of Samples


- Vanilla GANs do not allow controlled generation of samples
  - Example: Generate samples of a specific class
- How can we allow the condition to be incorporated into GAN modelling?



# Conditional Generation of Samples




Conditional GAN objective:

$$\min_G \max_D \left\{ -E_{x \sim X} \log D(x|y) - E_{z \sim \text{noise}} \log \left(1 - D(G(z|y))\right) \right\}$$

(y: class label.)

# Conditional Generation of Samples