Lecture-4 Ian

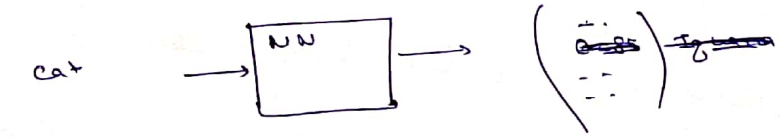
- Attacking NN with adversarial Examples
- Generative Adversarial Examples.

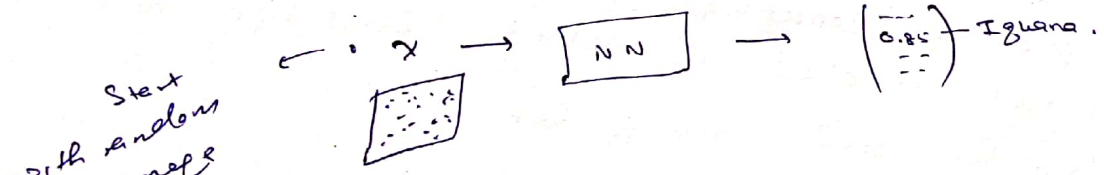
A. Attacking a network with adversarial
B. Defenses against adversarial examples
C. Why are neural networks vulnerable to Adversarial examples.

Paper - Intriguing properties of Neural Networks 2013 Szegedy
Explaining & Harnessing Adversarial Examples: 2015 Ian.

(A): Attacking a network with adversarial:

Goal: Given a network pretrained on Imagenet, find an input image that is not iguana but will be classified as an iguana.





Rephrasing what we want:

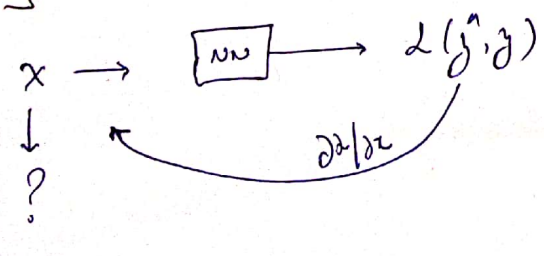
find x such that $\hat{y}(x) = y_{iguana} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

Loss function:

$$L(\hat{y}, y) = \frac{1}{2} \| \hat{y}(w, b, x) - y_{iguana} \|_2^2$$

↳ l_2 loss.

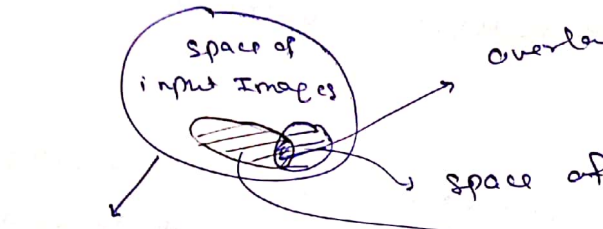
Optimize Image:



$$x = x - \alpha \frac{\partial L}{\partial x}$$

Q: Will the forged image look like Iguana?

- Maybe or not. But in terms of probability high chance that it don't



but we are probably in the black part that is not real Iguana.

Size of it?

$32 \times 32 \times 3 = 1024$

256

↳ size of an image.

↓

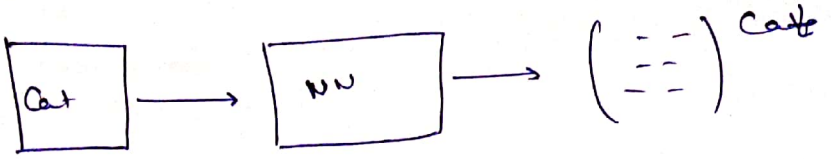
value of a pixel.

- ~~we are to in green black part~~

- This is dangerous when human sees cat but network see Iguana.

- Harder problems / consequences - face verification, violent content not detected as violent, Stop sign not Stop sign for self-driving car.

Goal: Given a network pretrained on Imagenet, find an input image that is cat but will be classify as an iguana.



Rephrasing:

Find x such that $\hat{y}(x) = y_{iguana} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

And $x \approx x_{cat}$.

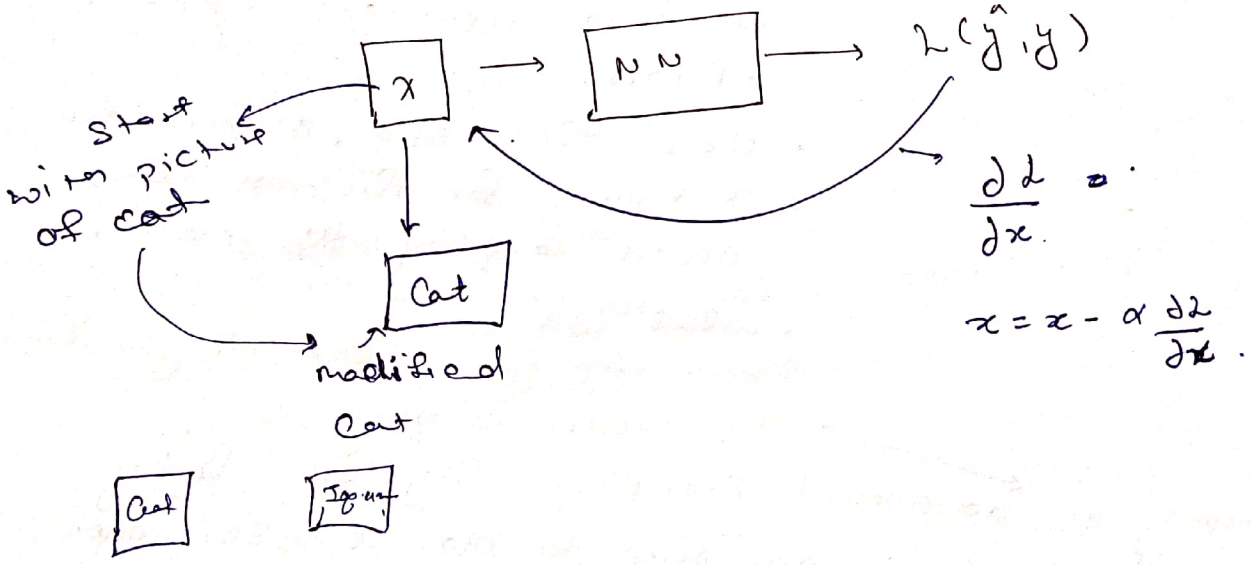
Loss:

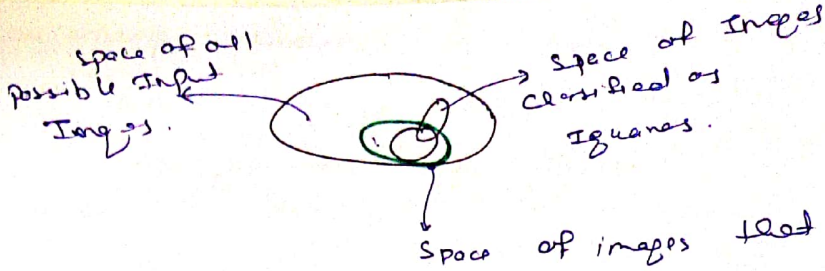
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2} ||\hat{y} - y||_2^2 + \lambda ||x - x_{cat}||_2^2$$

(x: variable.)

$\hat{y} = \hat{y}(w, b, x)$ (x: variable)

Optimize the Image:





dangerous. They look real to human but they might not be real.

(B) Defenses against Adversarial Examples:

Type of attacks

- non-targeted attacks.
- Targeted Attacks.

~~Defenses~~ Knowledge of attacker.

- White box ↑ access to model
- Black box.
  - when model is encrypted.

How would you attack when you have a black box?

- we cannot backpropagate but we can have numerical approximation of loss.
- change the input, observe the output. for this, we have access to query the model.
- ~~what will~~
- How will you attack if you don't have access to query.?

One property of adversarial Examples. They are highly transferable. what I am going to do is, say another model is animal classifier. I will build my own animal classifier, build a adversarial image on it. It is highly likely that it is an adversarial example for another model also.

Open Research Topic.

why this happen?
How to defend?

Paper:

Solution 1:

- Create a SafetyNet. This will allow only those that are real & not adversarial.
- Maybe we try to feed this also ~~as~~ but the constraint here is that we have to feed both at the same time.

Paper: SafetyNet - Detecting & Rejecting Adversarial examples Robustly.

Solution 2:

- Generate adversarial examples & train on it. — very high complexity.

Solution 3:

Adversarial training. $L_{new} = L(w, b, x, y) + \lambda L(w, b, \underset{\uparrow}{x_{adv}}, y)$ adversarial

↓

train on adversarial examples at the same time.

↳ Complexity - ?

- forward propagate x through the network to compute the first term.
- Generate x adversarial with the optimization ~~problem~~ process & forward propagate it to calculate the second term, then back propagate over the weights of network.

Super Costly.

Another:

Adversarial logit pairing:

$$L_{new} = L(w, b, x, y) + \lambda \| f(x; w, b) - f(x_{adv}; w, b) \|$$

Paper: Adversarial logit pairing.

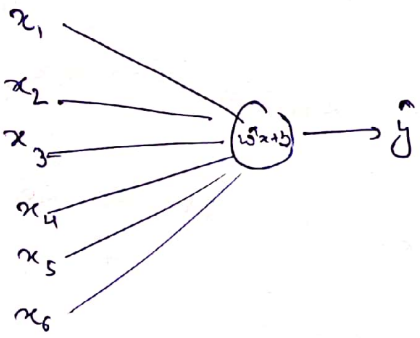
(C) Why are neural networks vulnerable to adversarial examples?

Why do adversarial examples exist?

Ian & his team argued that there are linear parts that are the cause of existence of adversarial examples. But, linear parts are what makes the training fast. like ReLu, also sigmoid, we focus on area that is linear.

We see by linear Regression:

forward prop: $\hat{y} = w^T x + b$.



Network has been trained & converged to

$w = [1, 3, -1, 2, 2, 3]$

$b = 0$

i) $x = \begin{bmatrix} -1 \\ 2 \\ 3 \\ -2 \end{bmatrix}$ $\Rightarrow \hat{y} = wx + b$

$= 1 - 3 - 2 + 0.6 - 6$

$= -4$

Q: How to change x into x^* such that $\hat{y}$ changes radically but $x^* \approx x$.?

Sol: $\frac{\partial \hat{y}}{\partial x}$ ~ impact on $\hat{y}$ on small changes of x.

$\frac{\partial \hat{y}}{\partial x} = w^T$

shape same as x, so w^T, not w.

$x^* = x + \epsilon w^T$

↳ value of perturbation.

$\hat{y}^* = w x^* + b = w x + \epsilon w w^T$

$b \to 0$

$w w^T \to \|w\|_2^2$

↳ always positive.

push $\hat{y}$ to larger value due to $+\epsilon \|w\|_2^2$, if $-\epsilon \|w\|_2^2$ then push $\hat{y}$ to lower value.

$x^* = x + \epsilon w^T$

$\epsilon \to 0.2$

$$\begin{bmatrix} 1 & +0.2 \\ -1 & +0.6 \\ 2 & -0.2 \\ 3 & +0.4 \\ 3 & +0.4 \\ -2 & +0.6 \end{bmatrix} = \begin{bmatrix} 1.2 \\ -0.4 \\ 1.8 \\ 0.4 \\ 3.4 \\ -1.4 \end{bmatrix}$$

↳ ~~dot product~~

$\hat{y}^* = w x^* = 1.2 + (-1.2) - 1.8 + 0.8 + 6.8 - 4.2$

$= 1.6$

Insights:

→ If w is large, then x^* is not similar to x.

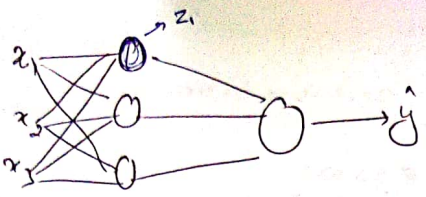
→ ↳ take sign (w)

→ As x grows in dimension the impact of $+\epsilon$ sign(w) increases. (Because $\Sigma w x^*$ grows) on $\hat{y}$

So we got it!. For images dimension is very high. Thus Adversarial examples are more prone to happen.

Fast gradient sign Method - formula to generate adversarial examples

$$x^* = x + \epsilon \text{ sign}\left(\nabla_x f(w, x, y)\right)$$



$$\frac{\partial L}{\partial x} = \sum \frac{\partial L}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial x}$$

↓ we want this high. Because to train the neuron corresponding to it, we need it high.

Similarly high.

$$\frac{\partial L}{\partial w_1^{[1]}} = \frac{\partial L}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial w_1^{[1]}}$$

Similarly to train w_1 we would need $\frac{\partial L}{\partial z_1^{[1]}}$ to be high.

The network that have high gradients & thus operating in the linear regime are more vulnerable to adversarial attacks, Because of above observation.

GANS

Do neural networks actually understand the data?

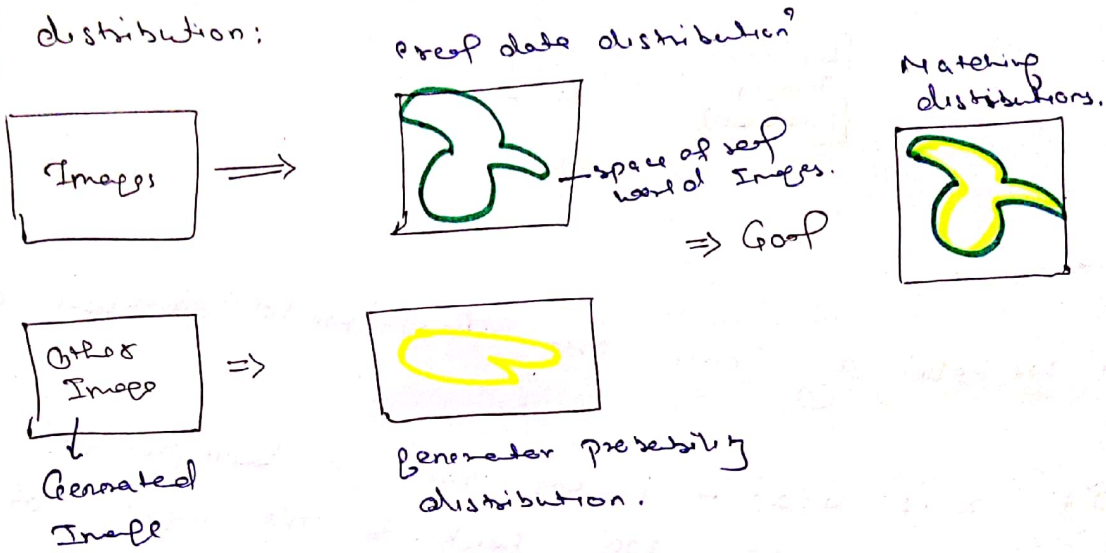
A. Motivation
B. G/D Game
C. Training GAN
D. NIQE Results
E. Evaluating GAN.

(A) Motivation: Endow computers with an understanding of world.

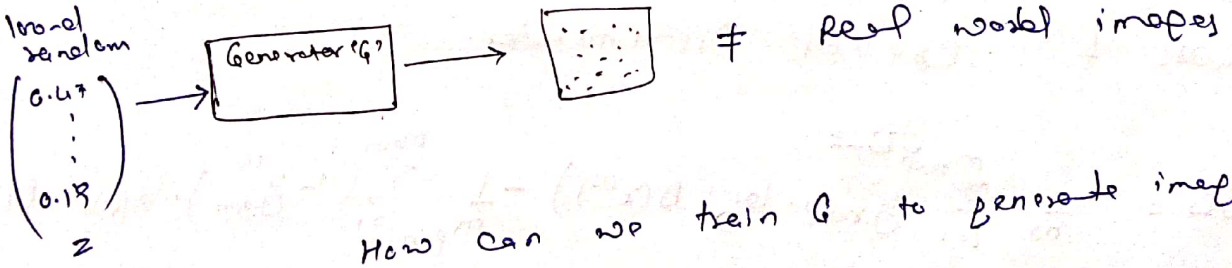
* Goal: Collect a lot of data, use it to train a model to generate similar data from scratch.

Intuition: no. of parameters of model << amount of data.

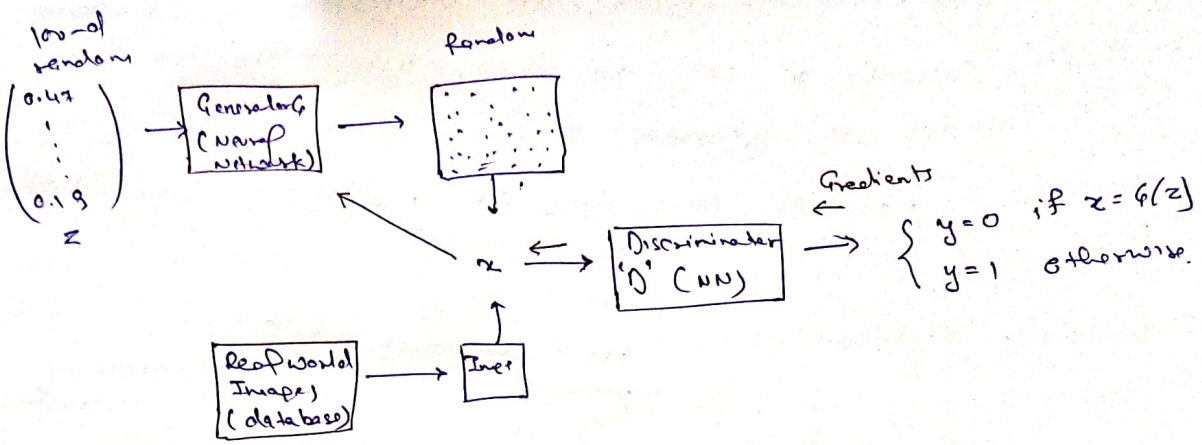
Probability distribution:



* We have so much amount of data. No. of parameters of the model is smaller than the amount of data. This is the intuition why generative networks can exist. Because there is too much of data, & there are not enough parameters to mimic this data. Because network have enough features to understand the salient features, but doesn't have enough to overfit everything.



How can we train G to generate images from true data distributions?



- We label any imgs from Real world database as one.

If x is G(z) - means we can back propagate all the way back to the input of the discriminator.

If x is from Real database - then there is no relation between z & x & hence b/w z & d.

- Run algorithm such as adam for both simultaneously
- There are many methods that are tried to train GAN properly.

Loss fn: Cost of Discriminator.

$$J_D = -\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} y_{real}^{(i)} \cdot \log\left(D(x^{(i)})\right) - \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \left(1 - y_{gen}^{(i)}\right) \cdot \log\left(1 - D(G(z^{(i)}))\right)$$

Cross entropy 1:
D should correctly label real data as 1

Cross entropy 2:
D should correctly label generated data as 0.

$y^{(i)}_{real} = 1$ always, $1 - y^{(i)}_{gen} = 1$ also. so we can remove them.

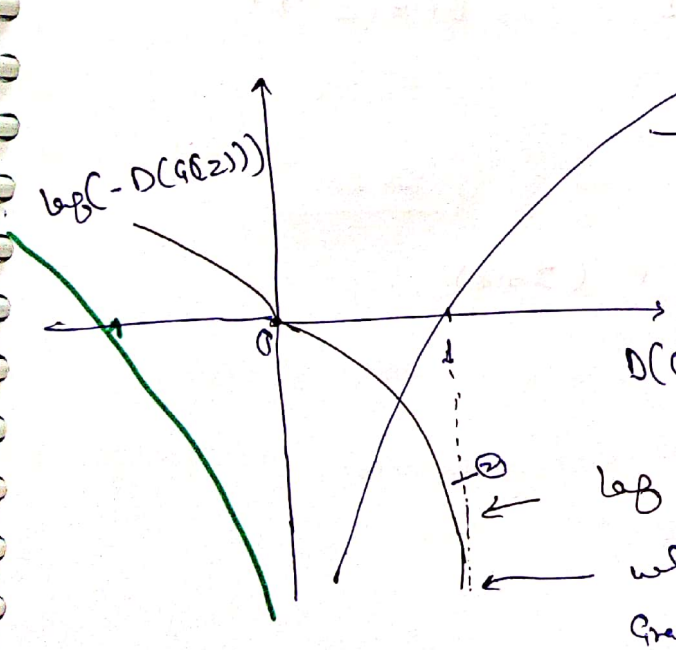
$$J_G = -J_D = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

Goal is to fool the Discriminator. If G managed to fool D, & D is very good then G is very good.

It's a game. If D is bad, we cannot say G is good.

We want D to go up to be very good & G to go up at the same time. Untill the equilibrium is reached at a certain point where D is always output one-half, like random probabilities. Because it cannot ~~sample~~ distinguish the samples coming from G versus real samples.

for generated images we want D to classify as one.



① which one is better ① or ②?
The closer question is - where are we in the beginning of training?
- obviously $D(G(z)) = 0$ in beginning. i.e discriminator can verify of G(z), so $\log(D(z)) = -\infty$. This ② one is better than ①.

$\log(1 - D(G(z)))$ = cost of Generator.

what is issue with this graph?
Gradient goes to very high at 1.

Thus:

But

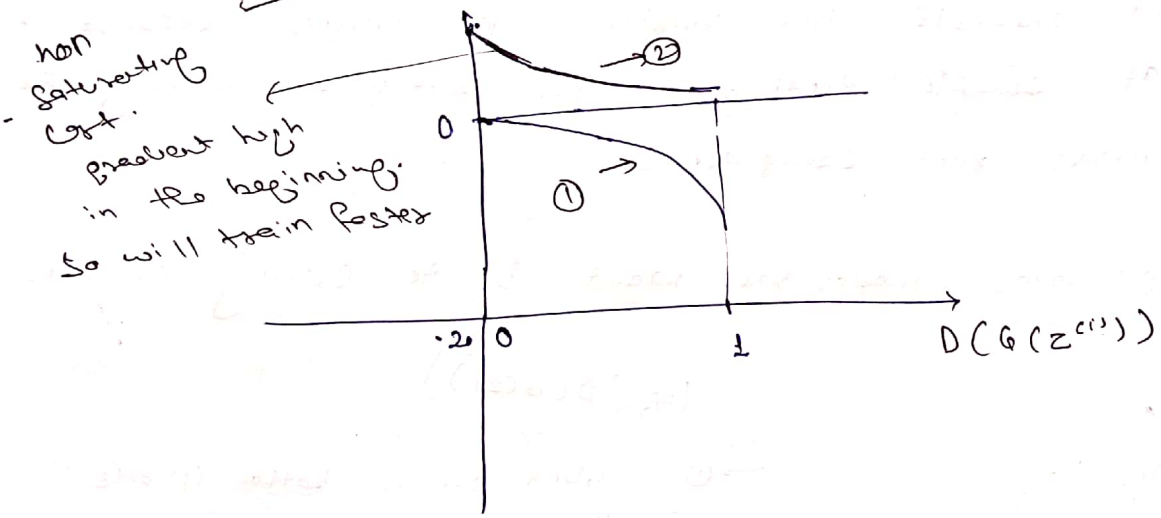
$$J(G) = -J(D) = \frac{-1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(1 - D(G(z^{(i)}))\right) \quad \text{—(1)}$$

Convert it to (1) form of $\log(D(G))$.

$$\min \log(1-x)$$
$$\equiv \max \log(x)$$
$$\equiv \min -\log(x)$$

now $$J_G = \frac{1}{m} \sum_{i=1}^{m_{gen}} -\log\left(D(G(z^{(i)}))\right)$$

$$= \min -\frac{1}{m} \sum_{i=1}^{m_{gen}} \log\left(D(G(z^{(i)}))\right) \quad \text{—(2)}$$



Paper: Are GAN created equal? (2018)

↓

Different loss functions.

(C) Training GAN:

- Modification to the cost function
- keep D up to date with respect to G. (K update for D/1 update for G)
- Use virtual BatchNorm.
- (not presented till now) One sided label smoothing.

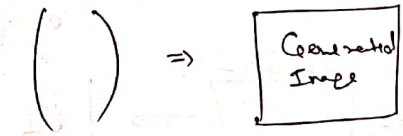
(D) Nice Results

1. Operation on code:

Code 1

man with specs.

Code 2

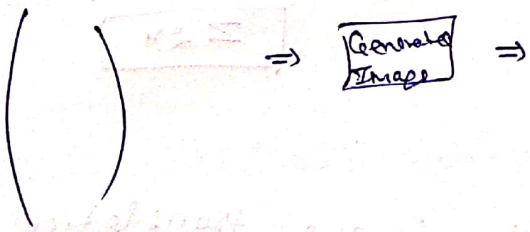


man without specs.

Code 3

woman without specs.

Code 1 - Code 2 + Code 3



woman with specs

fun!

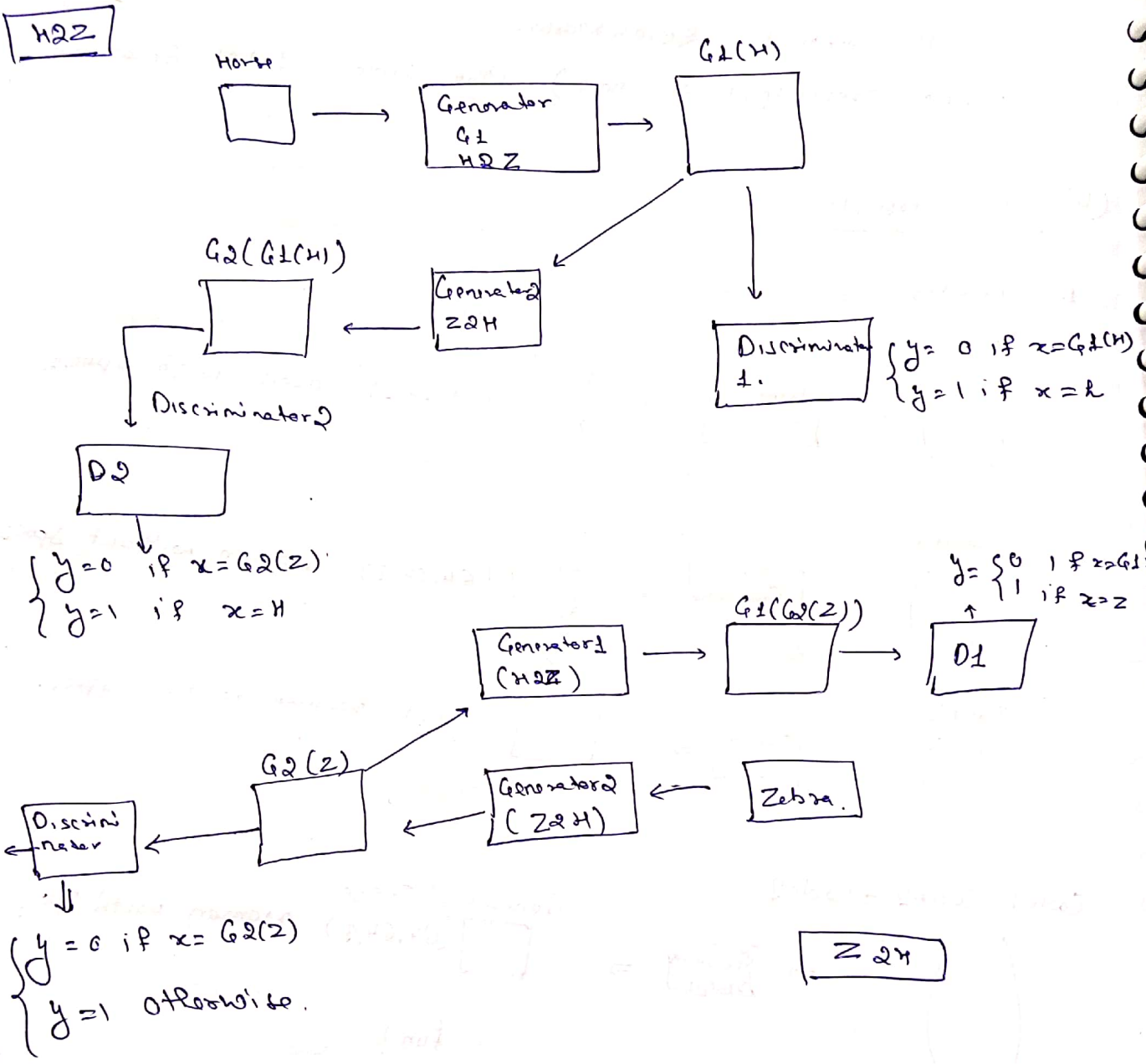
Paper: Unsupervised Representation learning with Deep generating adversarial networks.

Goal: Convert horses to zebras on images, vice-versa.

Data: Horses, Zebras - unpaired images.

Architecture: Cycle GAN.

H2Z



Paper: Unpaired image to image translation using cycle consistent Adversarial Networks

Loss function:

$$J^{D1} = -\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} \log\left(D1(H^{(i)})\right) - \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(1 - D1(G1(H^{(i)}))\right)$$

$$J^{G1} = -\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(D1(G1(H^{(i)}))\right)$$

$$J^{D2} = -\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} \log\left(D2(Z^{(i)})\right) - \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(1 - D2(G2(Z^{(i)}))\right)$$

$$J^{G2} = -\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log\left(D2(G2(Z^{(i)}))\right)$$

$$J = J^{D1} + J^{G1} + J^{D2} + J^{G2} + \lambda J^{cycle}$$

CycleGAN:

Papers:- face2Ramen.

:- face2Ramen using CycleGAN.

:- Unpaired Image-to-Image Translation using Cycle Consistent Adversarial Networks.

:. Pix2Pix. - generate image from drawings. train specifically for a domain.

:. Photo realistic single-Image super resolution using GAN.

↳ Higher Resolution image from lower resolution.

:- Privacy preserving generative model support clinical data sharing.

:- Learning beyond Human expertise with generative models for ~~healing~~ Dental restorations.

:- Unsupervised Cipher cracking using discrete GAN.

How to evaluate GAN?

- Human annotators.
- Inception Score (IS) $IS(G) = \exp\left(\mathbb{E}_{x \sim p_g}\left[KL\left(p(y|x) \,\|\, p(y)\right)\right]\right)$

↓ ↓

measure of image quality. measure of image diversity.

Papers: Are GANs Created Equal?

: Improved Techniques for training GANs.

Also Frechet Inception Distance (FID)