Generative Adversarial Network

Adapted from
https://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html

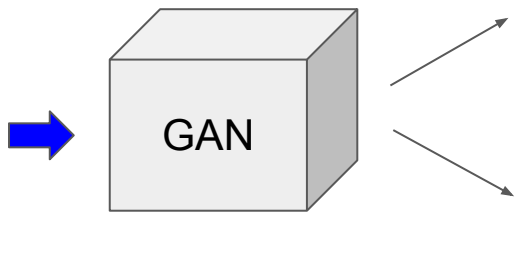
Introduction

Introduction

Purpose of GAN

Giving a vector (a list of values), GAN can produce a very realistic image or a series of words

$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$



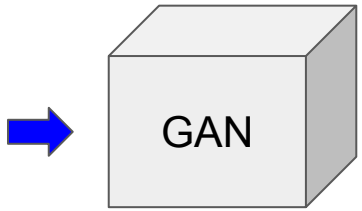
Good morning, I'm GAN.

Introduction

Purpose of GAN

Each element of the vector represents some features of the outputs

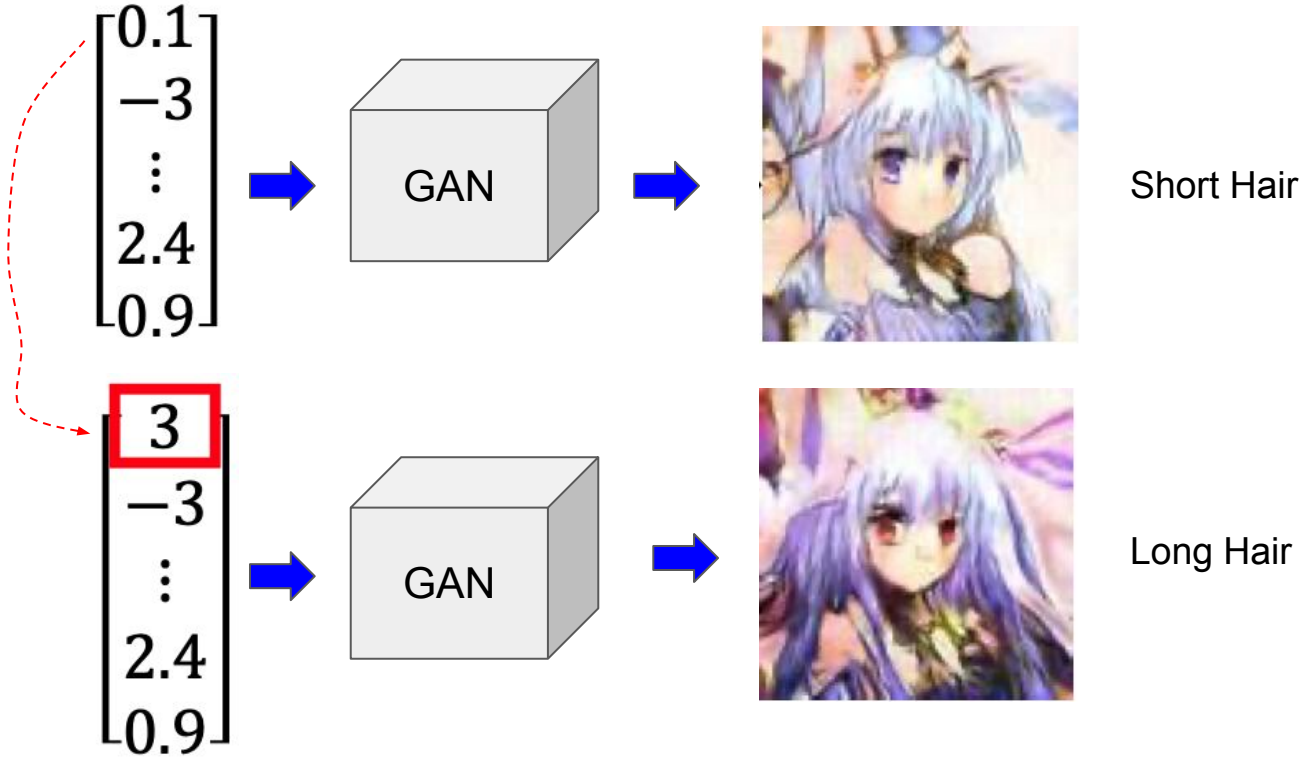
$$\begin{bmatrix} 0.1 \\ -3 \\ \vdots \\ 2.4 \\ 0.9 \end{bmatrix}$$



Introduction

Purpose of GAN

Each element of the vector represents some features of the outputs



For example, if we change the first element of the vector from 0.1 to 3, it gives us long hair in the output picture

Introduction

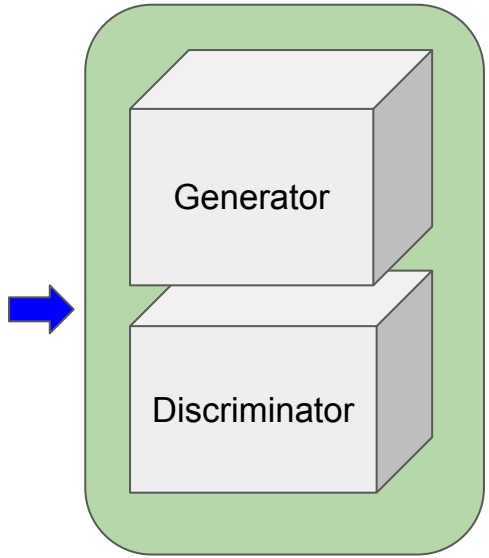
Basic structure of GAN

A **GAN** model contains two components:

- Generator
- Discriminator

Giving a vector (a list of values), GAN can produce a very realistic image or a series of words

$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$



GAN

Good morning, I'm GAN.

Introduction

Basic structure of GAN

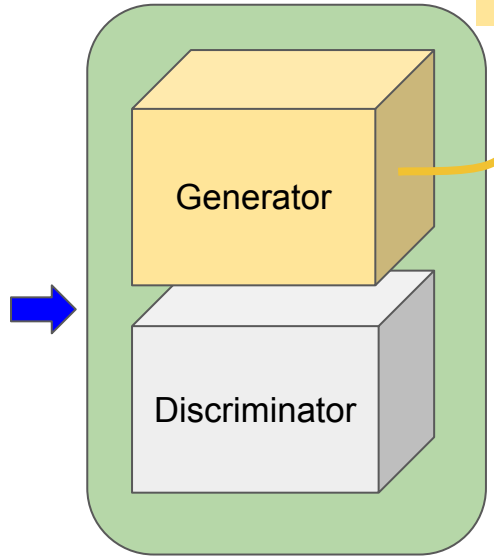
A **GAN** model contains two components:

- Generator
- Discriminator

Responsible for creating images

Giving a vector (a list of values), GAN can produce a very realistic image or a series of words

$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$



Good morning, I'm GAN.

GAN

Introduction

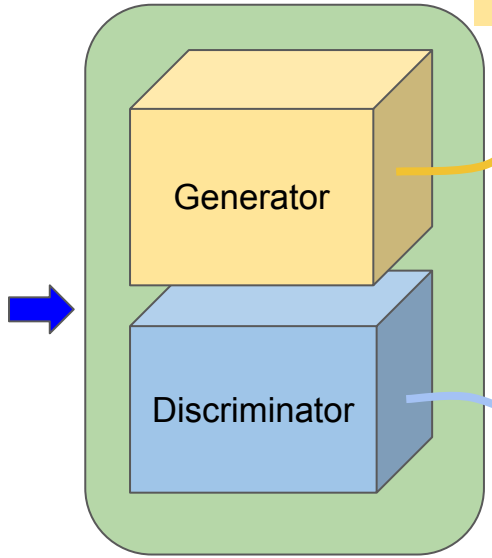
Basic structure of GAN

A **GAN** model contains two components:

- Generator
- Discriminator

Giving a vector (a list of values), GAN can produce a very realistic image or a series of words

$$\begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$



GAN

Responsible for creating images

Good morning, I'm GAN.

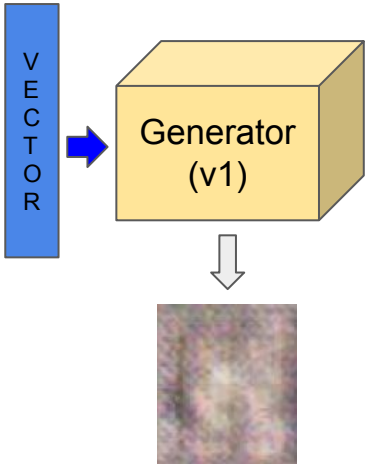
Responsible for evaluating the generated images

Basic GAN

Basic GAN

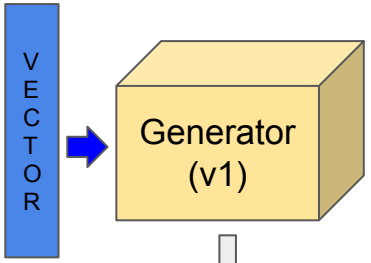
Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)



Basic GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)



Step 1b, the Discriminator (v1) will evaluate the similarity between the real images and the Generator produced image, and tell the differences

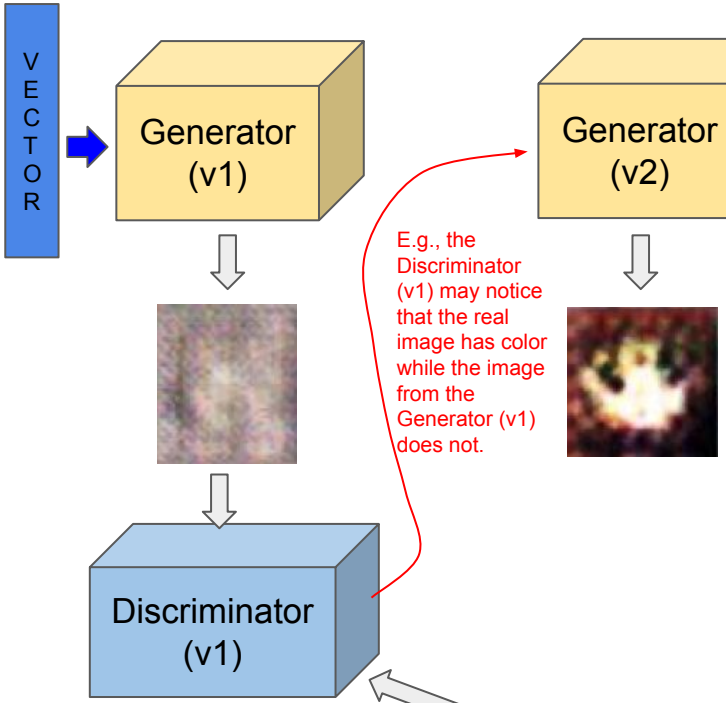
Real images from training dataset

Basic GAN

Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)



Step 1b, the Discriminator (v1) will evaluate the similarity between the real images and the Generator produced image, and tell the differences

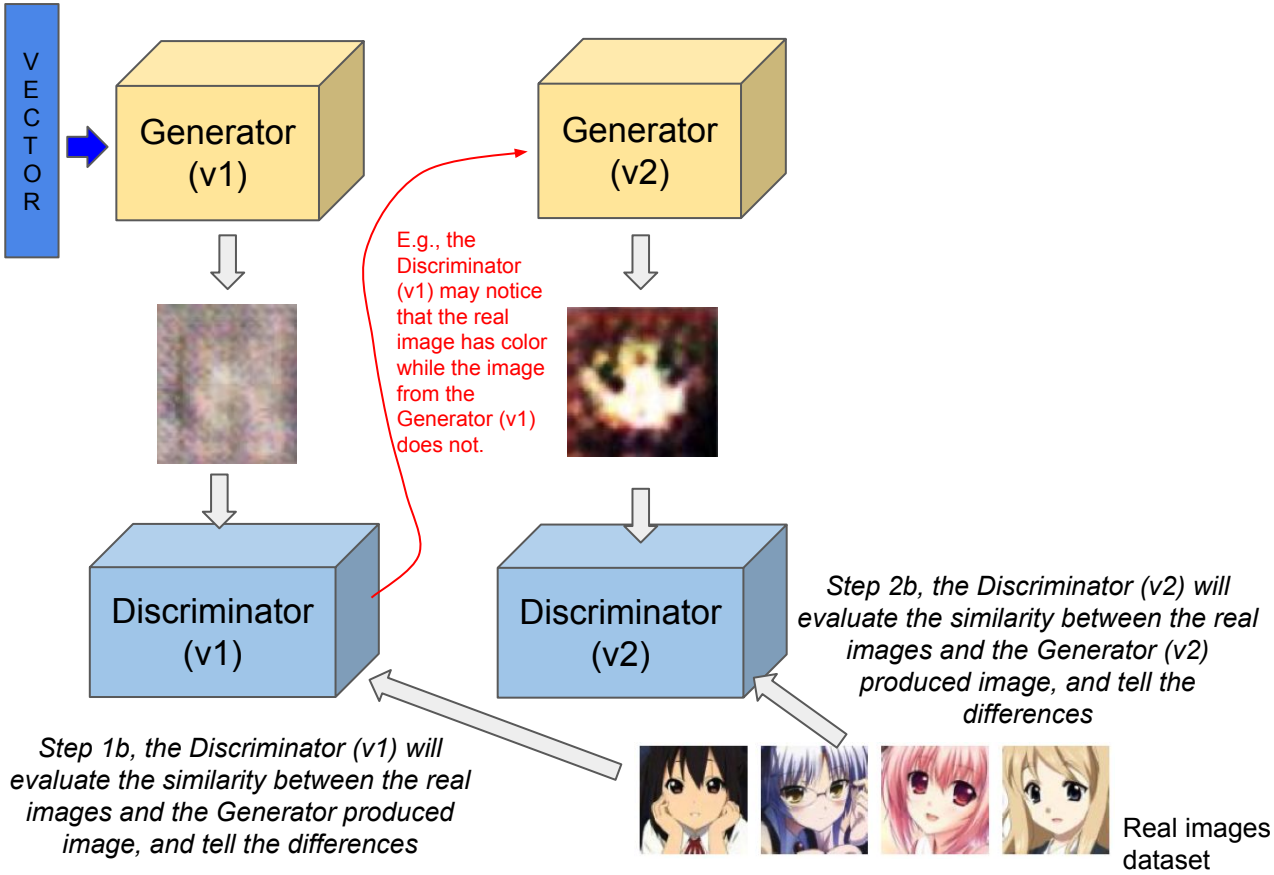
Real images from training dataset

Basic GAN

Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)



Step 2b, the Discriminator (v2) will evaluate the similarity between the real images and the Generator (v2) produced image, and tell the differences

Step 1b, the Discriminator (v1) will evaluate the similarity between the real images and the Generator produced image, and tell the differences

Real images from training dataset

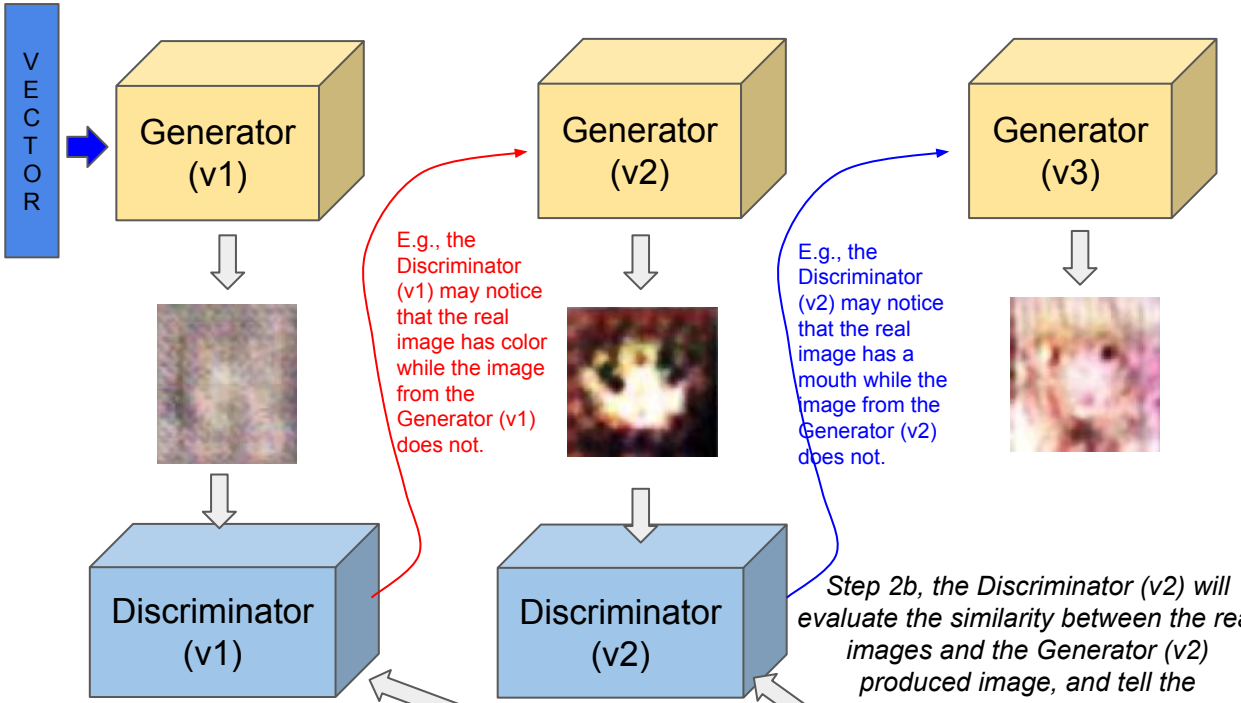
Basic GAN

Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

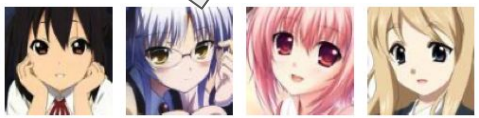
Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)

Step 3a, the Generator (v3) is set up based on v2, but incorporate the differences told by the Discriminator (v2)



Step 2b, the Discriminator (v2) will evaluate the similarity between the real images and the Generator (v2) produced image, and tell the differences

Step 1b, the Discriminator (v1) will evaluate the similarity between the real images and the Generator produced image, and tell the differences



Real images from training dataset

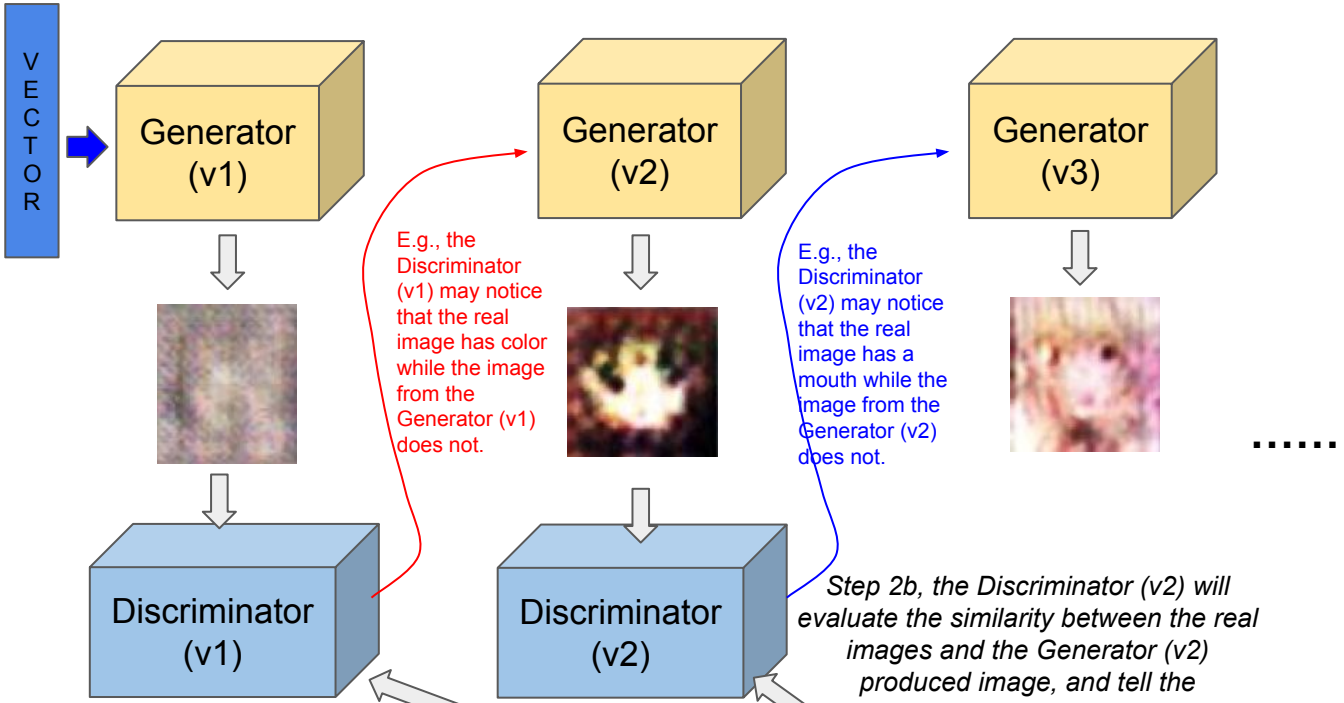
Basic GAN

Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)

Step 3a, the Generator (v3) is set up based on v2, but incorporate the differences told by the Discriminator (v2)



Generator and Discriminator evolves step-by-step iteratively, until we get a satisfied image

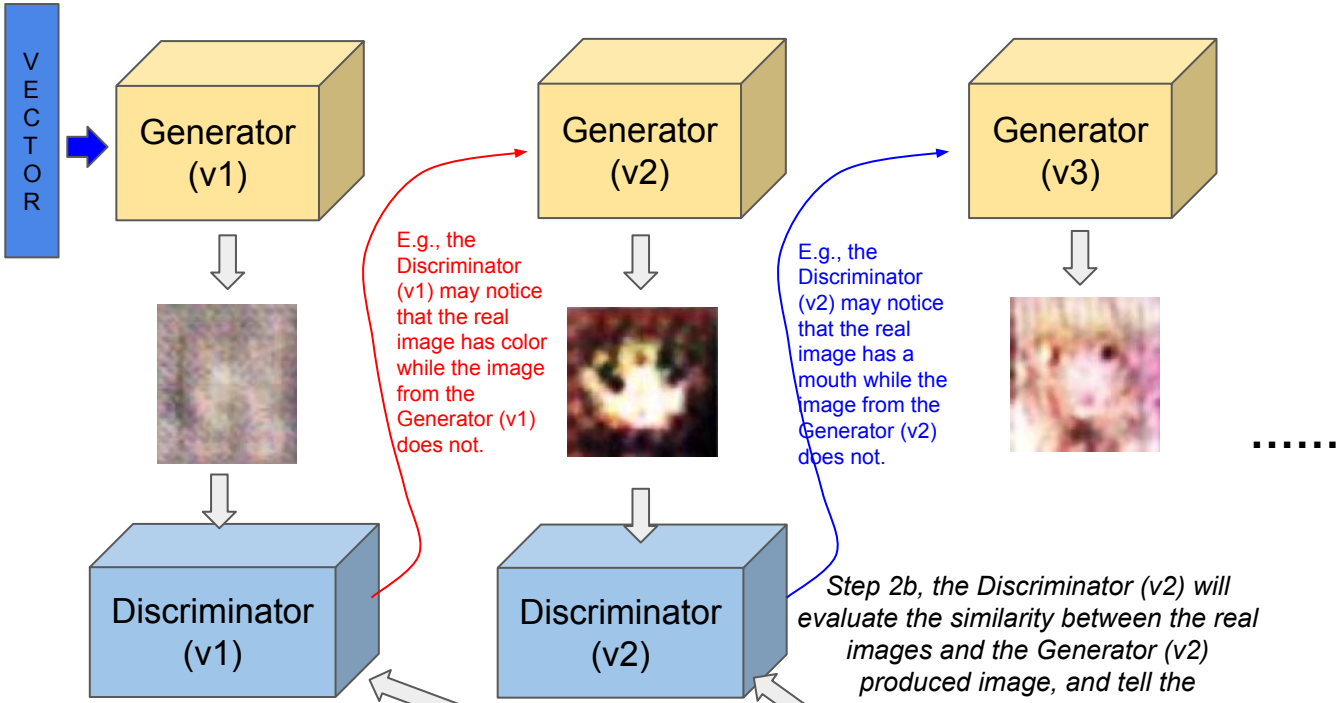
Basic GAN

Basic structure of GAN

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)

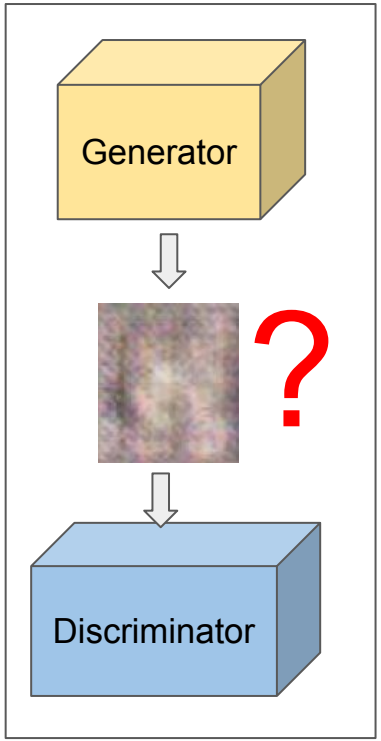
Step 3a, the Generator (v3) is set up based on v2, but incorporate the differences told by the Discriminator (v2)



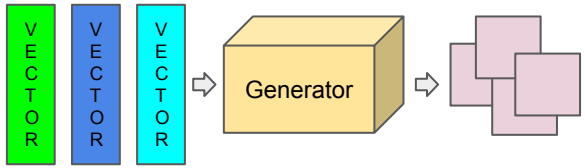
Generator and Discriminator evolves step-by-step iteratively, until we get a satisfied image

Basic GAN

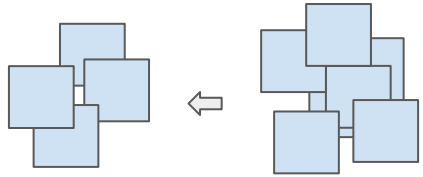
How each iterate works in GAN



Step 1: Fix the Generator, and train the discriminator (update its parameters)

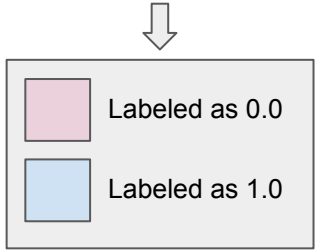


Giving the generator a bunch of random vectors, so it will produce a bunch of bad images



Some images are sampled from real images

From the training dataset, there are many real images



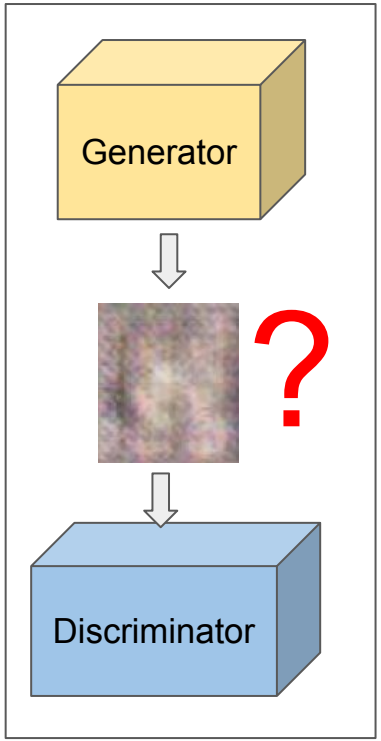
So the Discriminator parameters need to be updated to make:

- Images from generator are labelled as 0.0
- Real images are labelled as 1.0

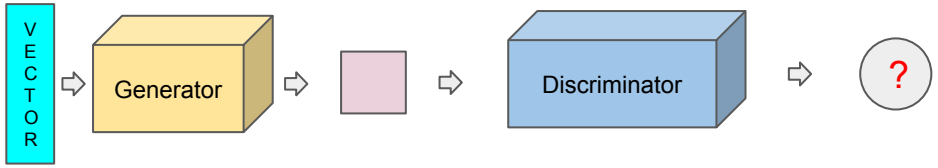
We can consider this as a regression problem to solve.

Basic GAN

How each iterate works in GAN



Step 2: Fix the Discriminator, and train the Generator (update its parameters)



Giving the generator a vector, through the generator it will produce an image

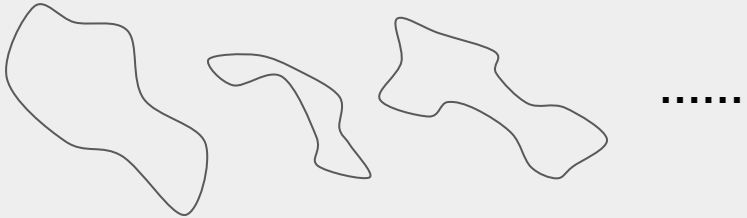
Then this image will be evaluated by the discriminator (we just updated it in Step 1)

Discriminator will give the image a score, the higher the score, the more closer to the real image

So we will tune the parameters of the generator to make the score as high as possible

An Example

We have many real images like the below:



.....

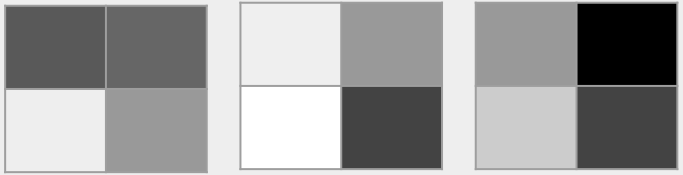
The matrix to contain the image is only 2x2:

So the image is represented as something like below

.....

So the 2x2 image is sort of "backlash + diagonal" …

We also have many images (2x2) only contains noises, e.g.,

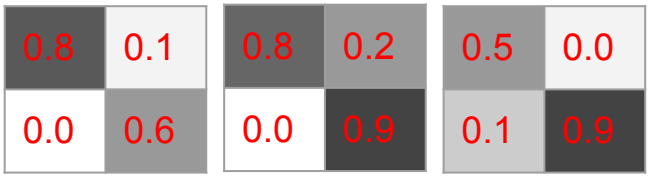


.....

So the noisey 2x2 image is not "backlash + diagonal" …

Some values are assigned to the 2x2 images

Real

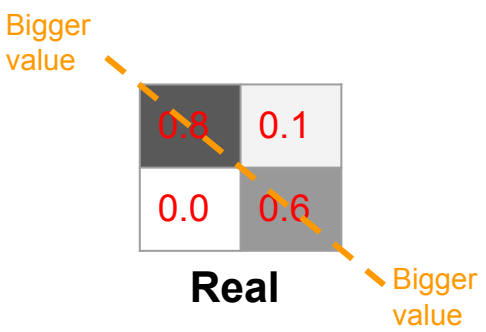


.....

Fake

.....

Step 1: Building discriminator



For this particular example, the real image has bigger values along the diagonal axis, while for the fake image, anything could happen …

In order to distinguish “real” and “fake”, we can **learn** the criteria in the model, e.g., we hope to have sth like below:

x1	x2
x3	x4

x1 + x4 - x2 - x3

The higher value from the index, the higher possibility that the image is real. We can set a threshold to say if an image is real or not !

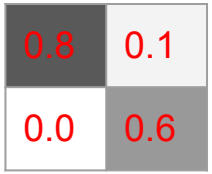
So for this example, we have

Real 0.8 + 0.6 - 0.1 - 0.0 = 1.3

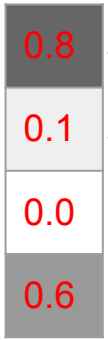
Fake 0.8 + 0.6 - 0.8 - 0.0 = 0.6

Step 1: Building discriminator

Assuming that we have a one neuron, single layer neural network



Real



Vectorize the image

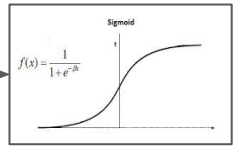
w1
w2
w3
w4

bias

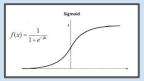
wb

(w1*x1 + w4*x4 + w2*x2 + w3*x3) + wb*bias

Then the output of neuron will go through an activation function (e.g., Sigmoid function)



F [(w1*x1 + w4*x4 + w2*x2 + w3*x3) + wb*bias]

1.2 ⇨  ⇨ 0.76

In discriminator there is a **threshold** (e.g., 0.5, in this case), 0.76 is larger than 0.5, so the discriminator will say it is a real image

w1=1, w2=-1, w3=-1, w4=1

wb=-1, bias=1.0

E.g., how to distinguish real & fake is learnt as "weights" in the neural network:

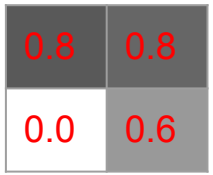
x1	x2
x3	x4

x1 + x4 - x2 - x3

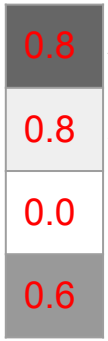
(1*0.8 + 1*0.6+ (-1)*0.1 + (-1)*0.0) + (-1)*1.0 = 1.2

Step 1: Building discriminator

Similarly, if a fake image goes through the neural network



Fake



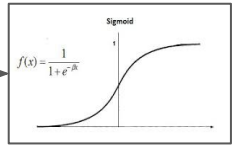
Vectorize the image

w1
w2
w3
w4
bias
wb

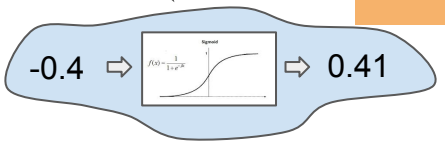
(w1*x1 + w4*x4 + w2*x2 + w3*x3) + wb*bias =

(1*0.8 + 1*0.6+ (-1)*0.8 + (-1)*0.0) + (-1)*1.0 = -0.4

Then the output of neuron will go through an activation function (e.g., Sigmoid function)



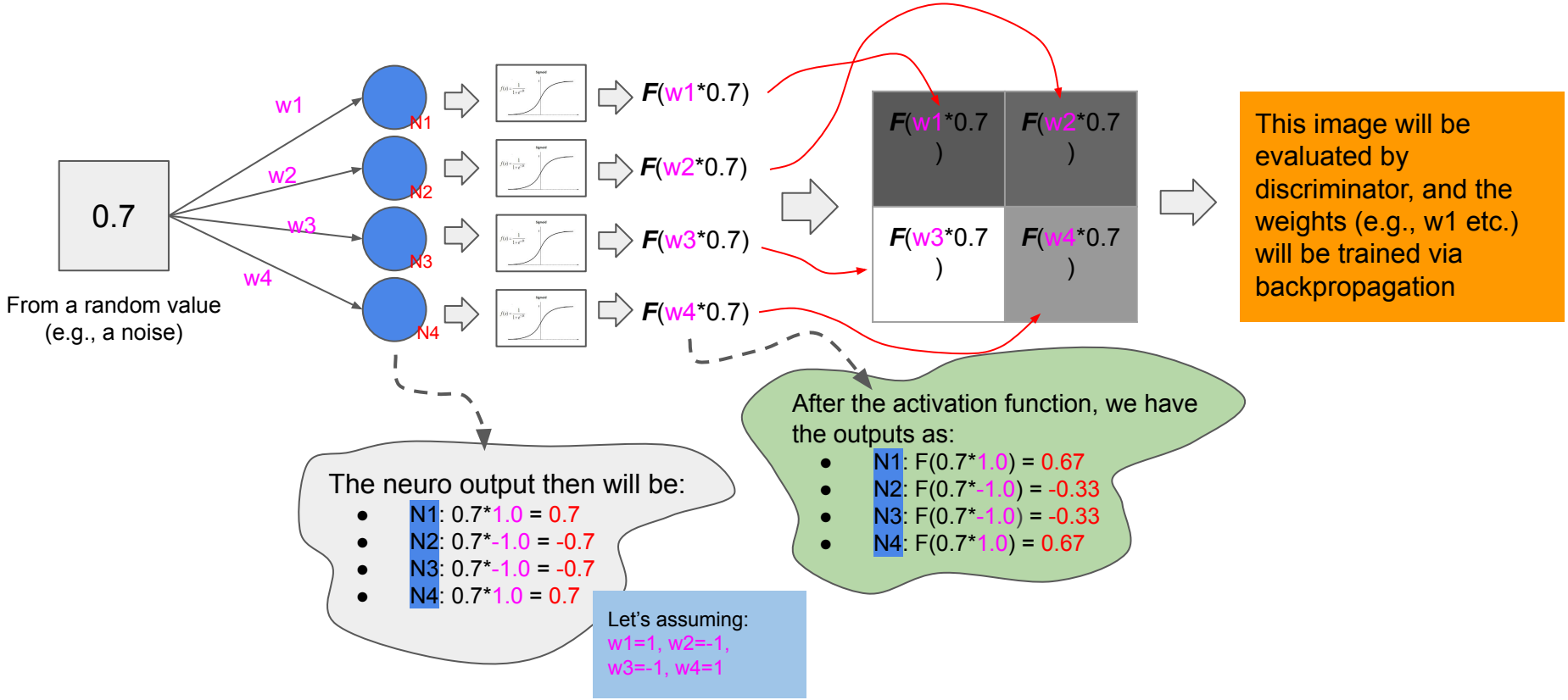
F [(w1*x1 + w4*x4 + w2*x2 + w3*x3) + wb*bias]



In discriminator there is a **threshold** (e.g., 0.5, in this case), 0.41 is less than 0.5, so the discriminator will say it is a fake image

Step 2: Building generator

Assuming that we have a 4 neurons (since the output will have 4 pixels), single layer neural network

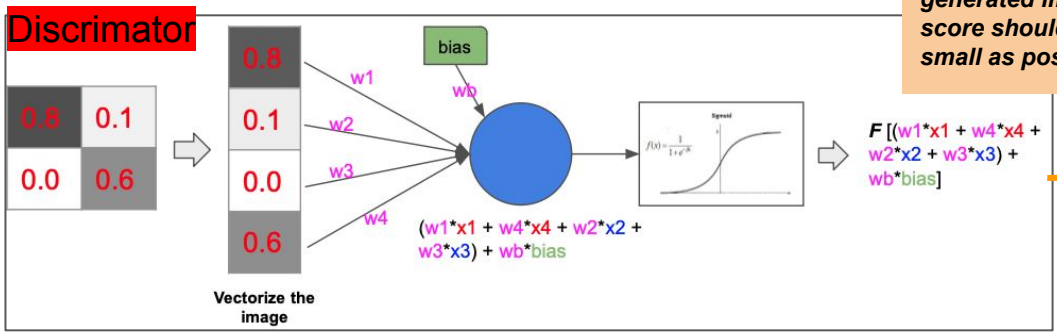


An overview for this simple model

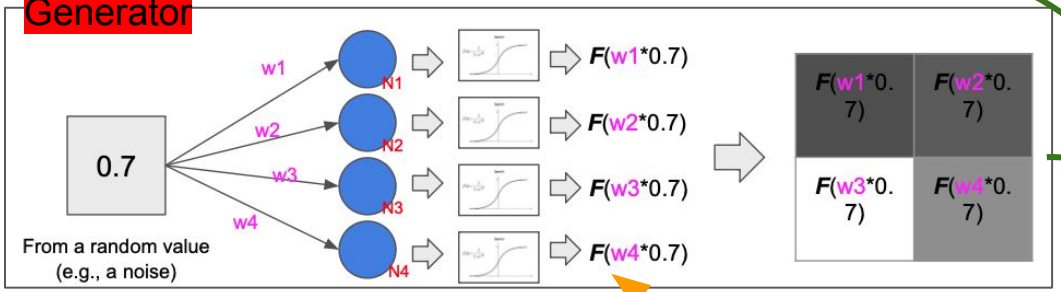
Real image

Step 1: train Discriminator, and hope to make "score" between real image and the generated one as big as possible ***(so for the generated image, the score should be as small as possible)***

Discrimator



Generator



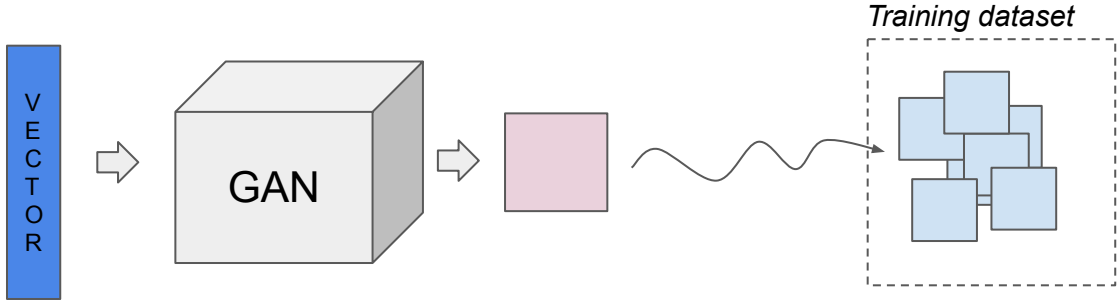
GAN generated image

Step 2: train Generator, and hope to make the "score" for the generated image as big as possible

Conditional GAN

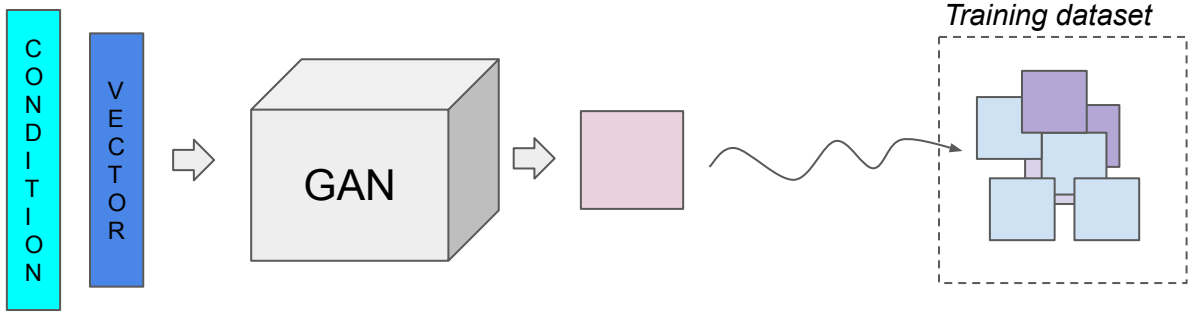
Conditional GAN

Traditional GAN



The user input a random vector, the model will produce an image similar to the ones in training dataset

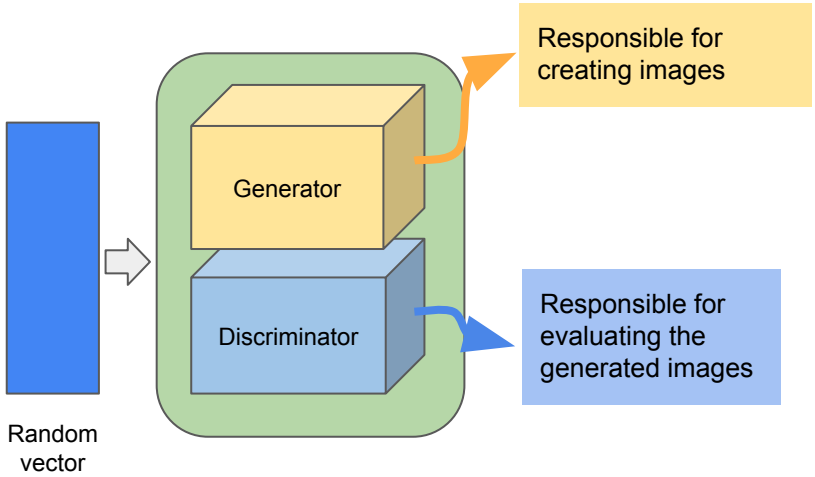
Conditional GAN



The user input a random vector and a conditional vector, the model will produce an image similar to the ones (with the required conditions) in training dataset

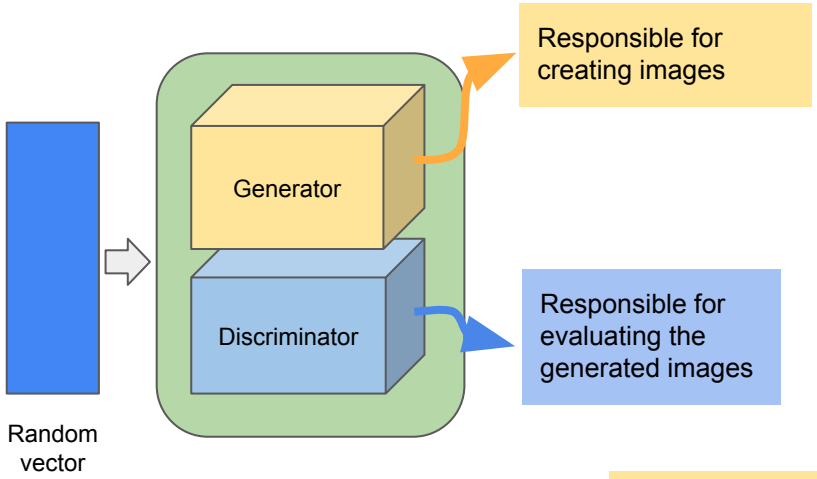
Conditional GAN

Traditional GAN

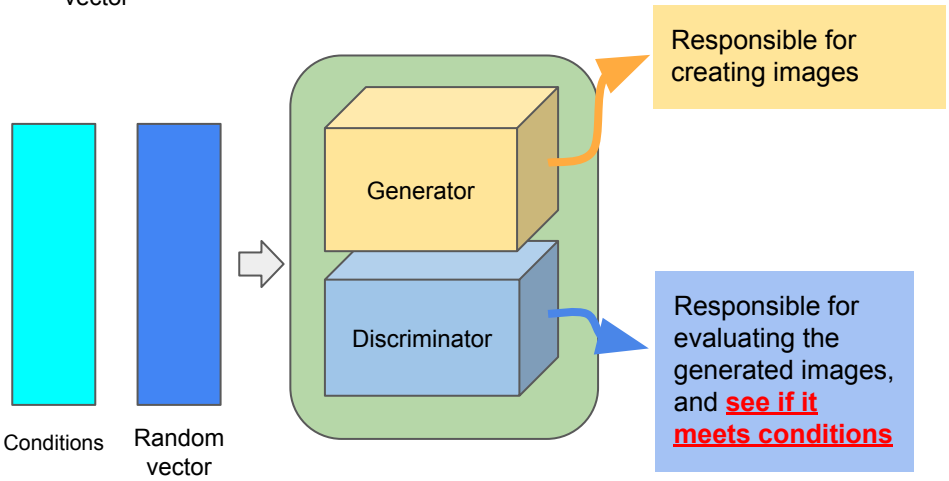


Conditional GAN

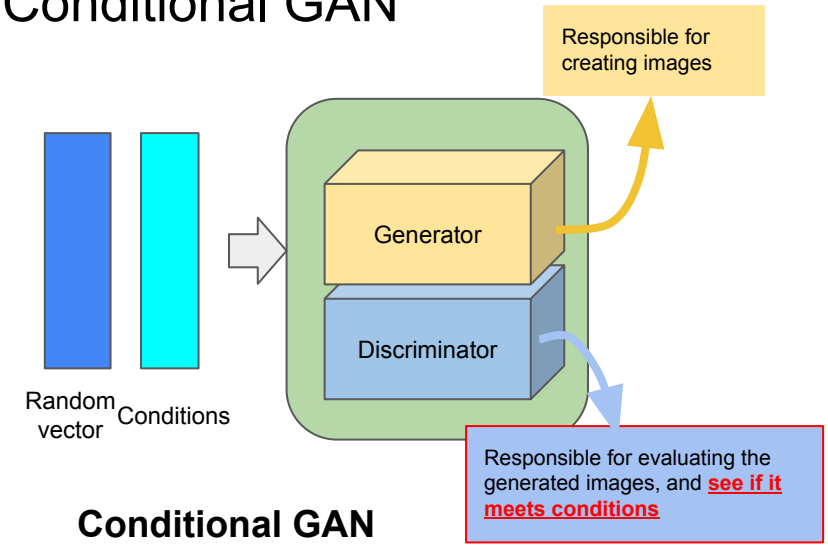
Traditional GAN



Conditional GAN

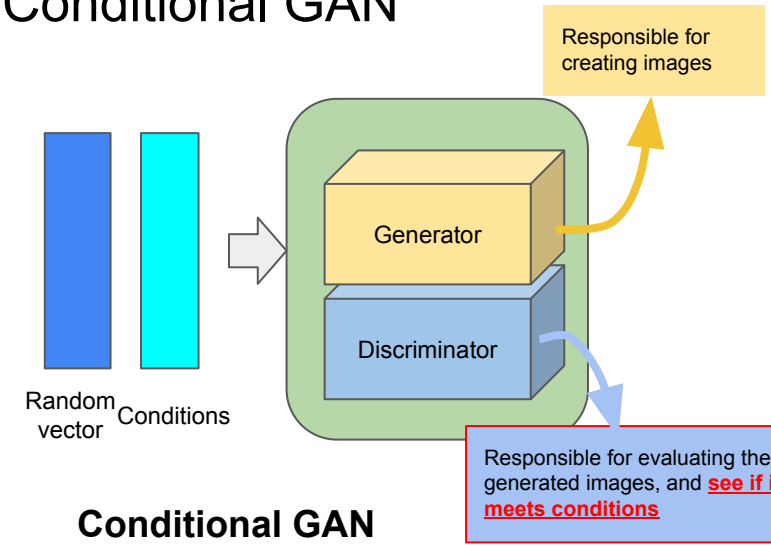


Conditional GAN

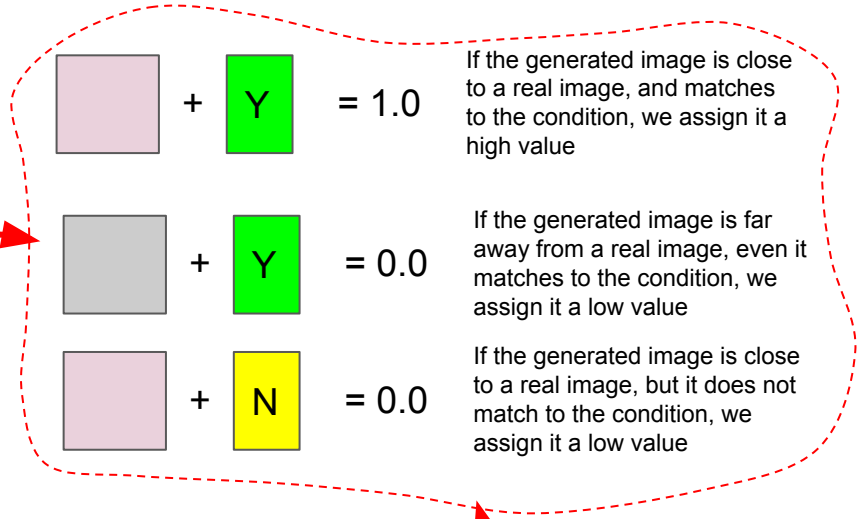


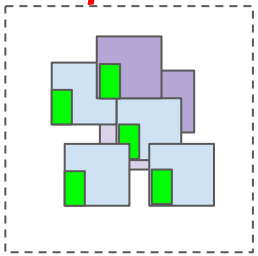
Conditional GAN

Conditional GAN



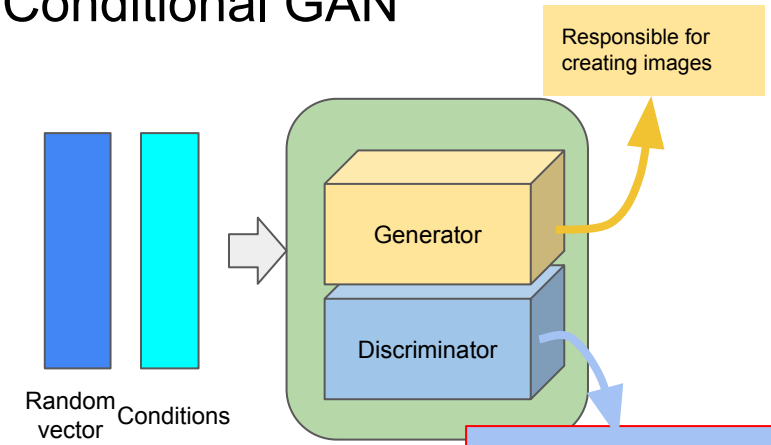
Conditional GAN



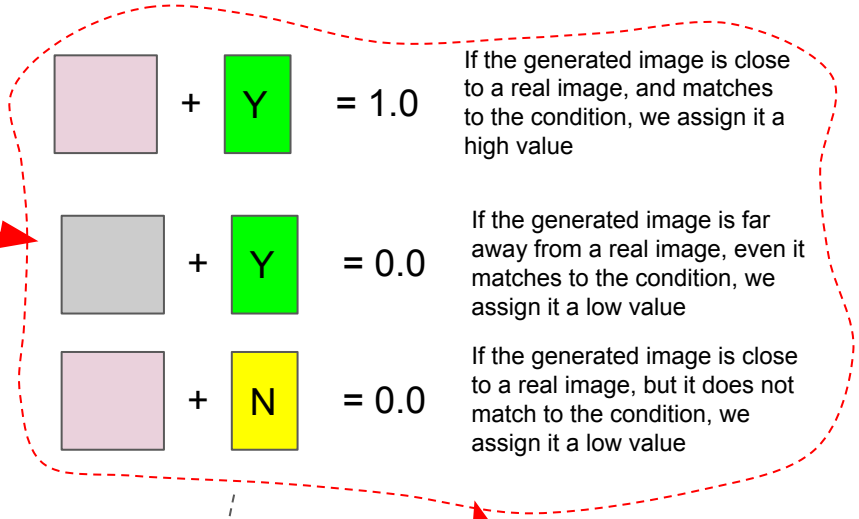


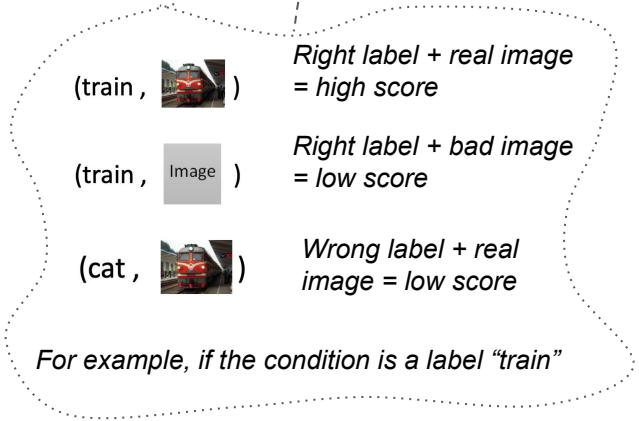
Training dataset has (labels + images)

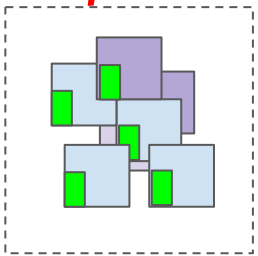
Conditional GAN



Conditional GAN





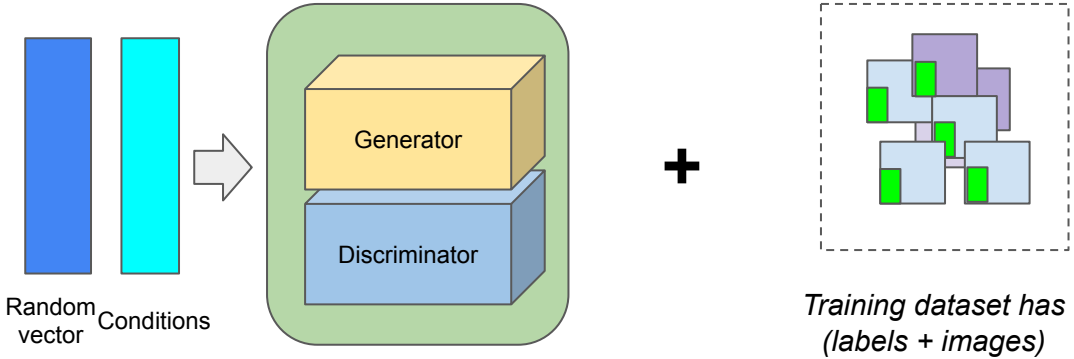


Training dataset has (labels + images)

Unsupervised GAN

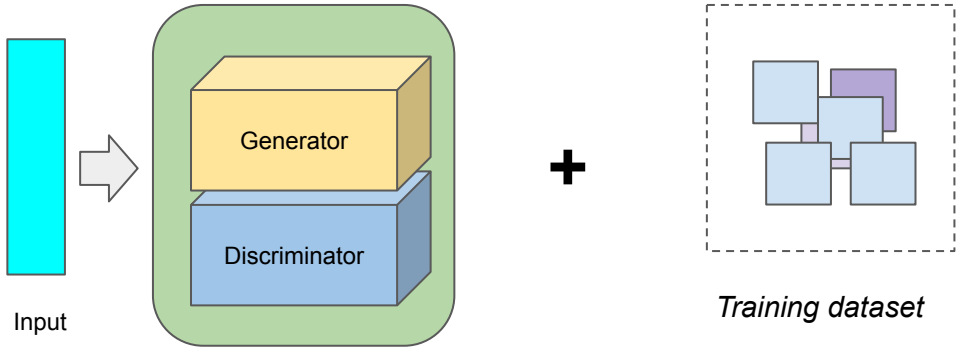
Unsupervised GAN

Conditional GAN



For conditional GAN, training dataset must be labelled

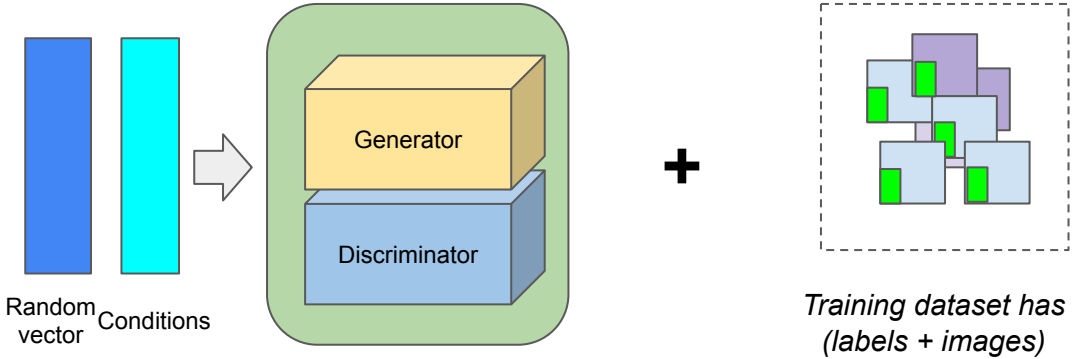
Unsupervised GAN



For Unsupervised GAN, training dataset does not have labels. When you give the input, the GAN model will output something similar to the ones in the Training dataset automatically

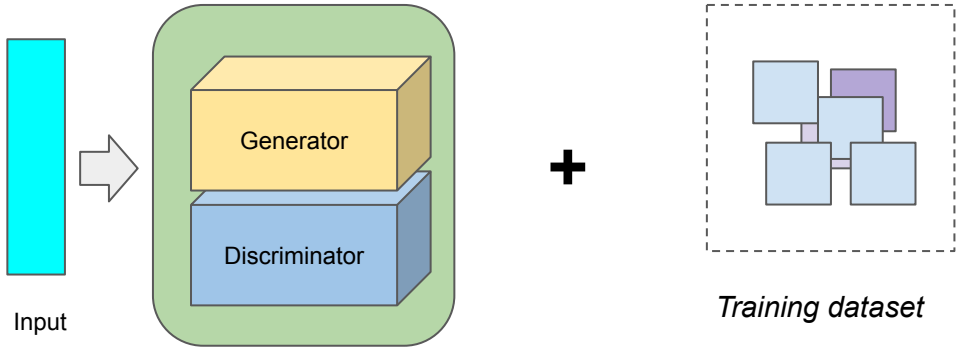
Unsupervised GAN

Conditional GAN



For conditional GAN, training dataset must be labelled

Unsupervised GAN

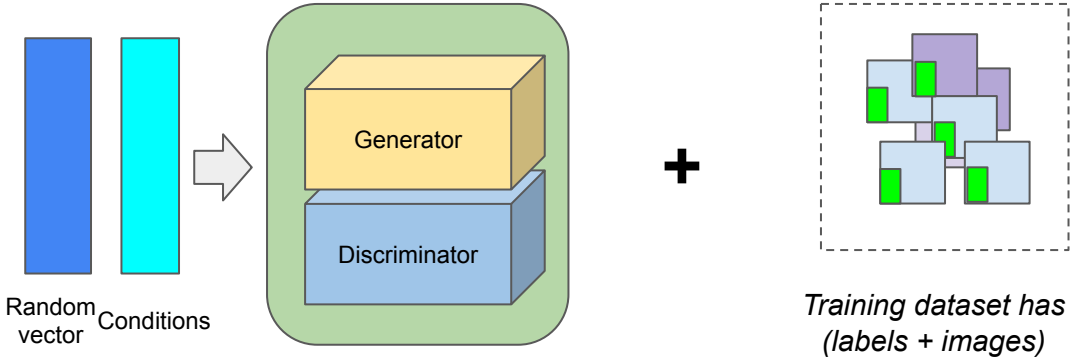


So in the unsupervised GAN, there are two bunch of data: data1 and data2. The machine will learn how to convert data1 to data2 automatically.

For Unsupervised GAN, training dataset does not have labels. When you give the input, the GAN model will output something similar to the ones in the Training dataset automatically

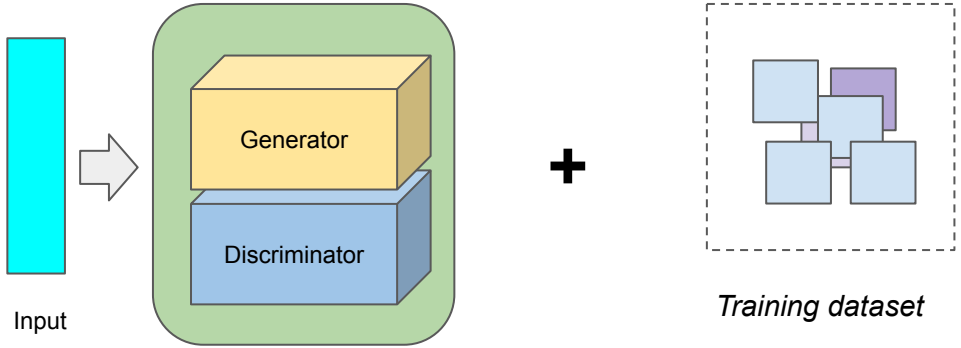
Unsupervised GAN

Conditional GAN



For conditional GAN, training dataset must be labelled

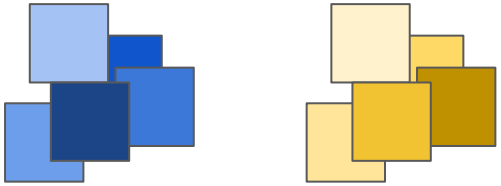
Unsupervised GAN



So in the unsupervised GAN, there are two bunch of data: data in domain 1 and domain 2. The machine will learn how to convert data1 to data2 automatically.

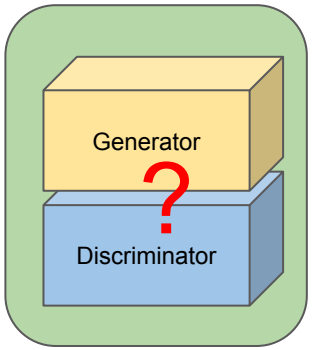
For Unsupervised GAN, training dataset does not have labels. When you give the input, the GAN model will output something similar to the ones in the Training dataset automatically

Unsupervised GAN



Domain 1 Domain 2

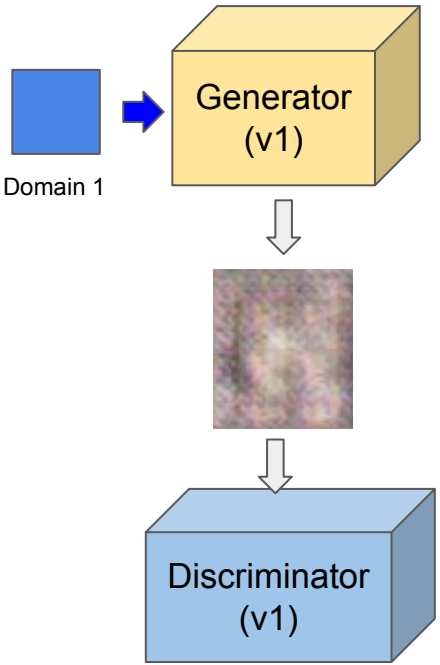
In unsupervised GAN's training dataset, we have many data in Domain 1, and many data in Domain 2



Through the training process, we hope to find parameters for the GAN model that we can convert data from Domain 1 to Domain 2 (essentially we want to find the relationship between Domain 1 and Domain 2)

Unsupervised GAN Method 1

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)



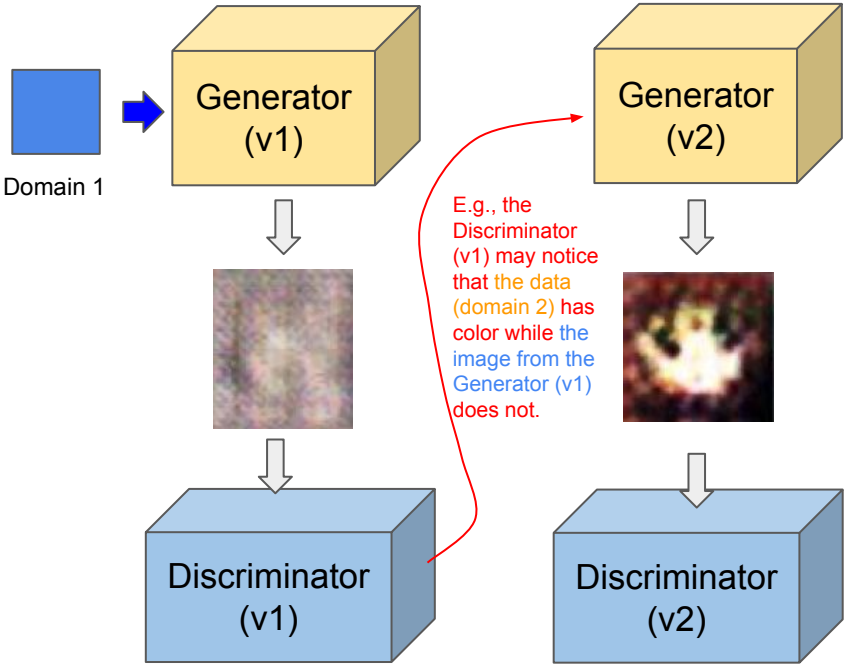
Step 1b, the Discriminator (v1) will evaluate the similarity between the data from domain 2 and the Generator produced image, and tell the differences

Unsupervised GAN

Method 1

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)



Step 2b, the Discriminator (v2) will evaluate the similarity between the data from domain 2 and the Generator (v2) produced image, and tell the differences

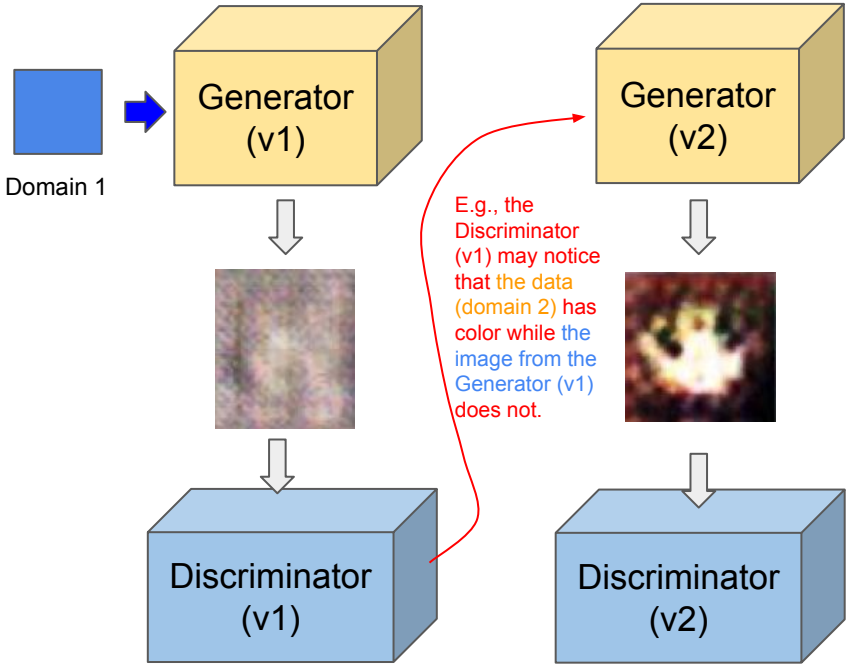
Step 1b, the Discriminator (v1) will evaluate the similarity between the data from domain 2 and the Generator produced image, and tell the differences

Unsupervised GAN

Method 1

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)



Step 1b, the Discriminator (v1) will evaluate the similarity between the data from domain 2 and the Generator produced image, and tell the differences

Step 2b, the Discriminator (v2) will evaluate the similarity between the data from domain 2 and the Generator (v2) produced image, and tell the differences

.....

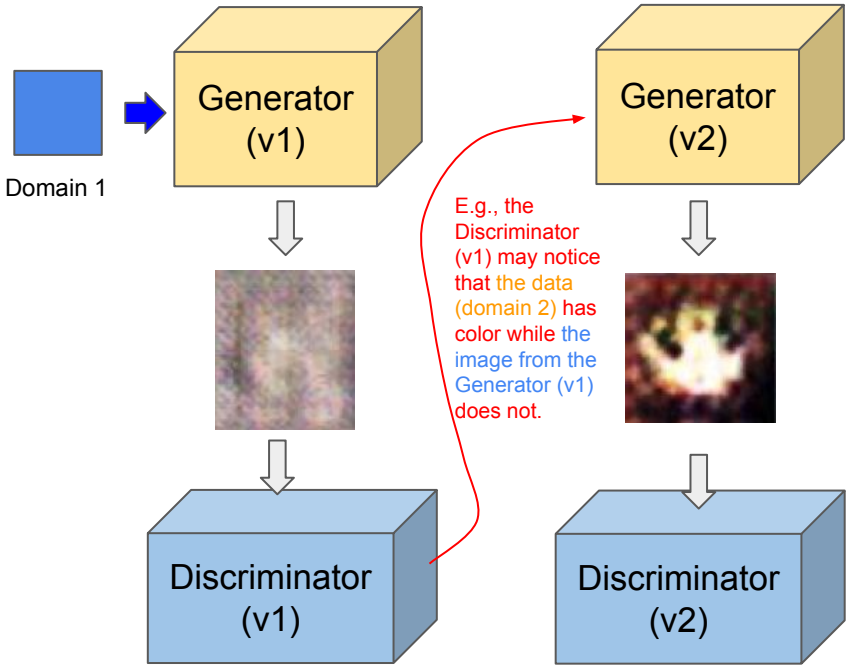
Generator and Discriminator evolves step-by-step iteratively, until generator can convert the data from Domain 1 to sth like the one in Domain 2

Unsupervised GAN

Method 1

Step 1a, the Generator (v1) has random parameters, so it will create a very bad image (many noises)

Step 2a, the Generator (v2) is set up based on v1, but incorporate the differences told by the Discriminator (v1)



Step 1b, the Discriminator (v1) will evaluate the similarity between the data from domain 2 and the Generator produced image, and tell the differences

Step 2b, the Discriminator (v2) will evaluate the similarity between the data from domain 2 and the Generator (v2) produced image, and tell the differences

The problem for this method is that:

If there are many interactive steps, the output produced by the GAN model could have little connection to the original input data (e.g., if Domain 1 and Domain 2 has big difference, over interaction, the features of Domain1 could loss gradually)

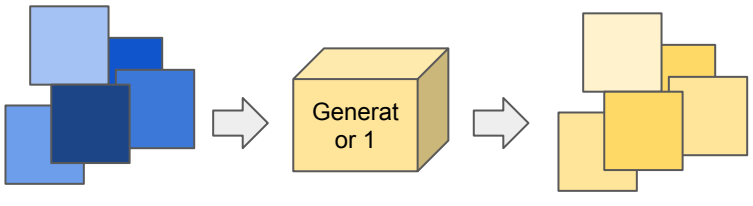
.....

Generator and Discriminator evolves step-by-step iteratively, until generator can convert the data from Domain 1 to sth like the one in Domain 2

Unsupervised GAN Method 2: cycle GAN

The purpose is to convert data from Domain 1 to Domain 2, but we have two generators:

First, it is a regular generator to convert Domain 1 to Domain 2



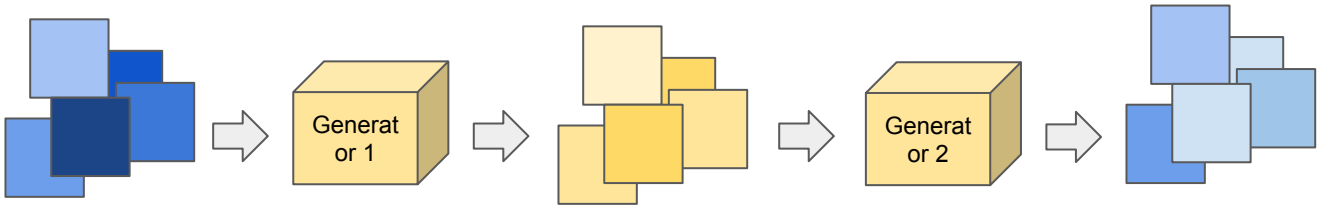
Generator 1
(convert Domain 1 data to Domain 2)

Unsupervised GAN Method 2: cycle GAN

The purpose is to convert data from Domain 1 to Domain 2,
but we have two generators:

First, it is a regular generator to convert Domain 1 to Domain 2

Then we use the generated data, and convert it back to Domain 1 using another generator



Domain 1

Generator 1
(convert Domain 1 data to Domain 2)

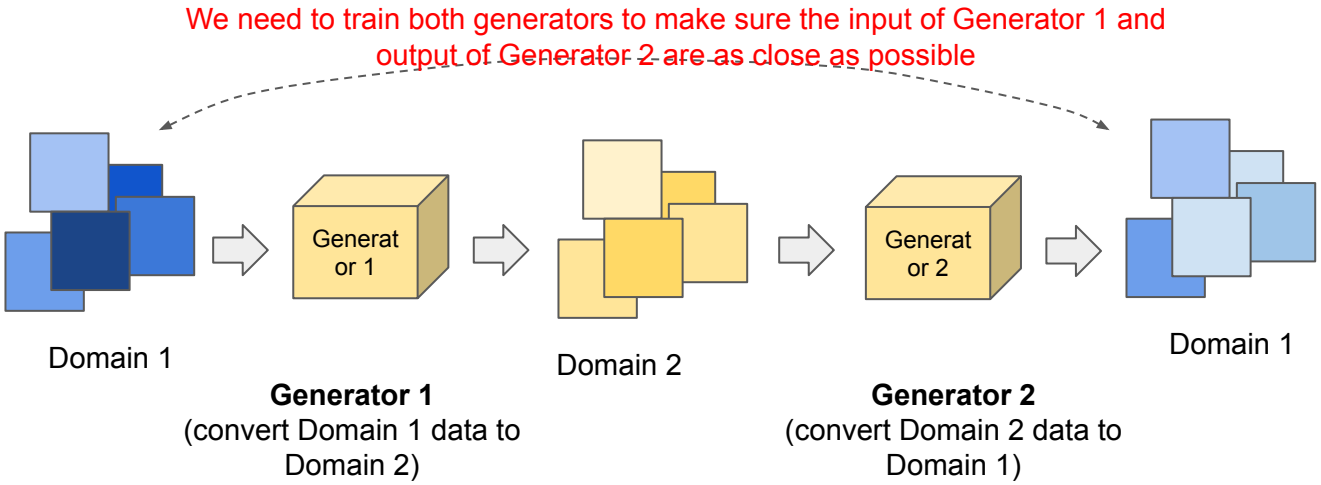
Domain 2

Generator 2
(convert Domain 2 data to Domain 1)

Domain 1

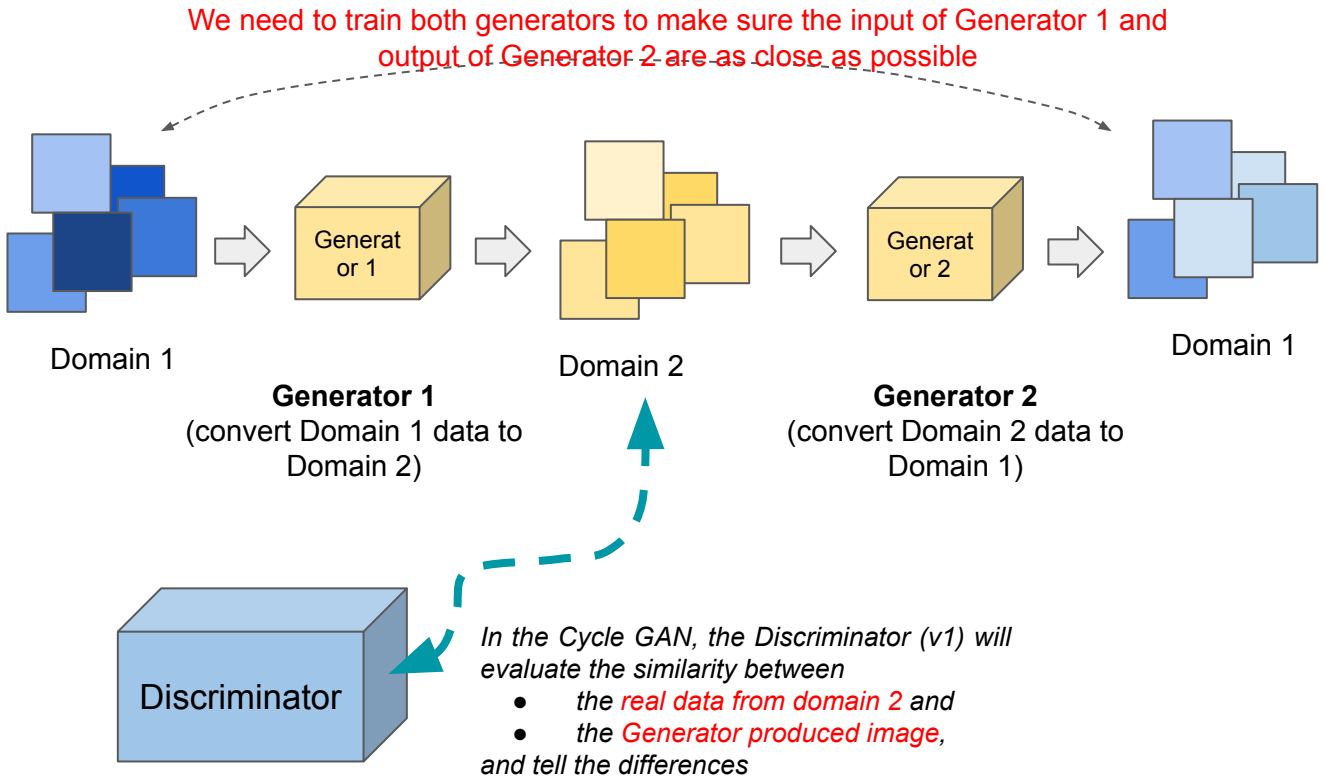
Unsupervised GAN Method 2: cycle GAN

The purpose is to convert data from Domain 1 to Domain 2,
but we have two generators:



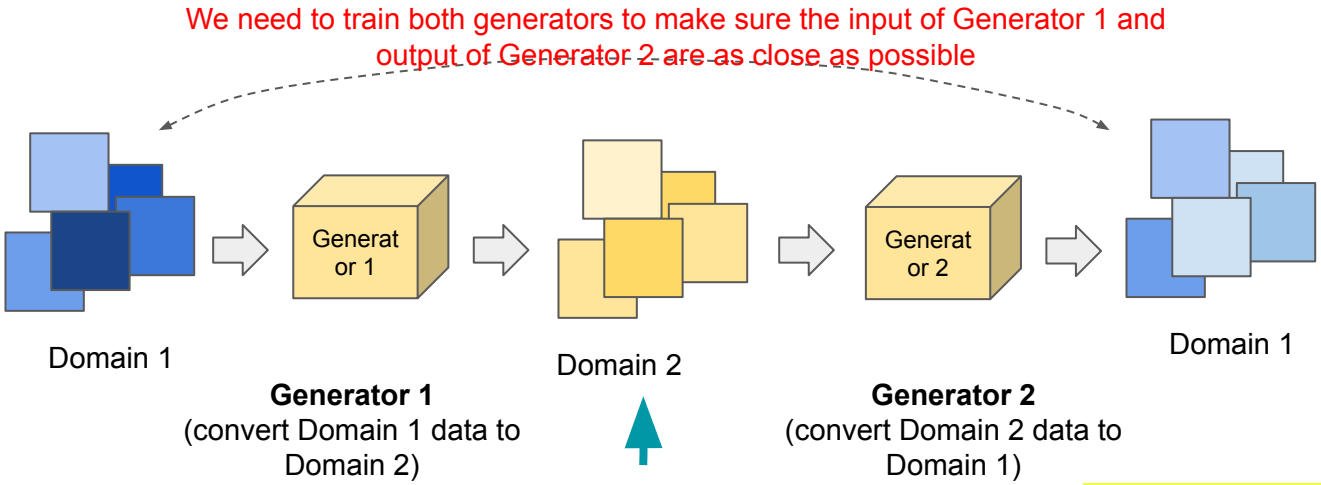
Unsupervised GAN Method 2: cycle GAN

The purpose is to convert data from Domain 1 to Domain 2,
but we have two generators:



Unsupervised GAN Method 2: cycle GAN

The purpose is to convert data from Domain 1 to Domain 2,
but we have two generators:



Compared to method 1, the method 2 can better keep the feature from the Domain 1 (e.g., the "Domain 2" produced from "Generator 1" must contain enough information of the input "Domain 1", otherwise it cannot be reverted back using "Generator 2")