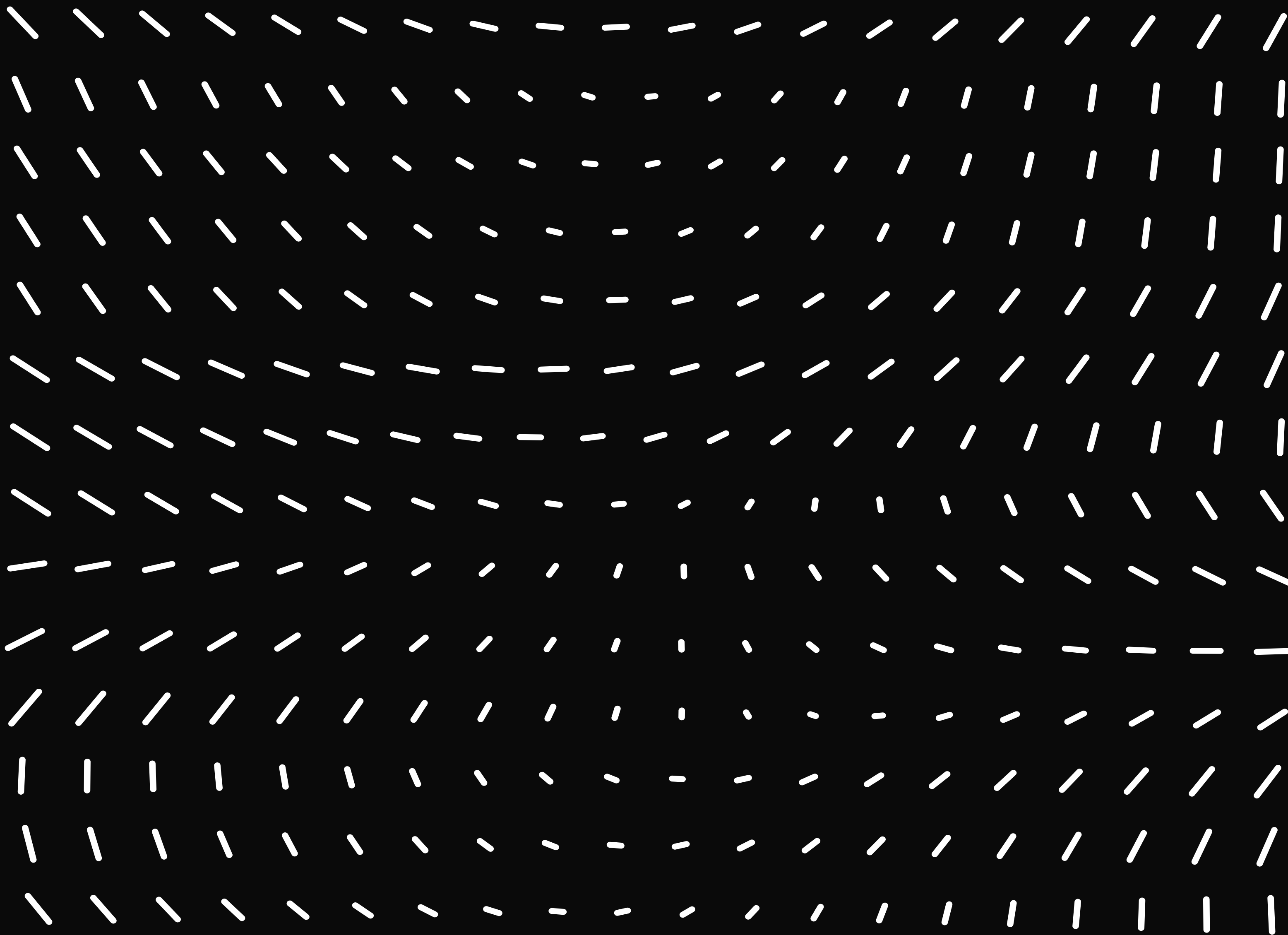
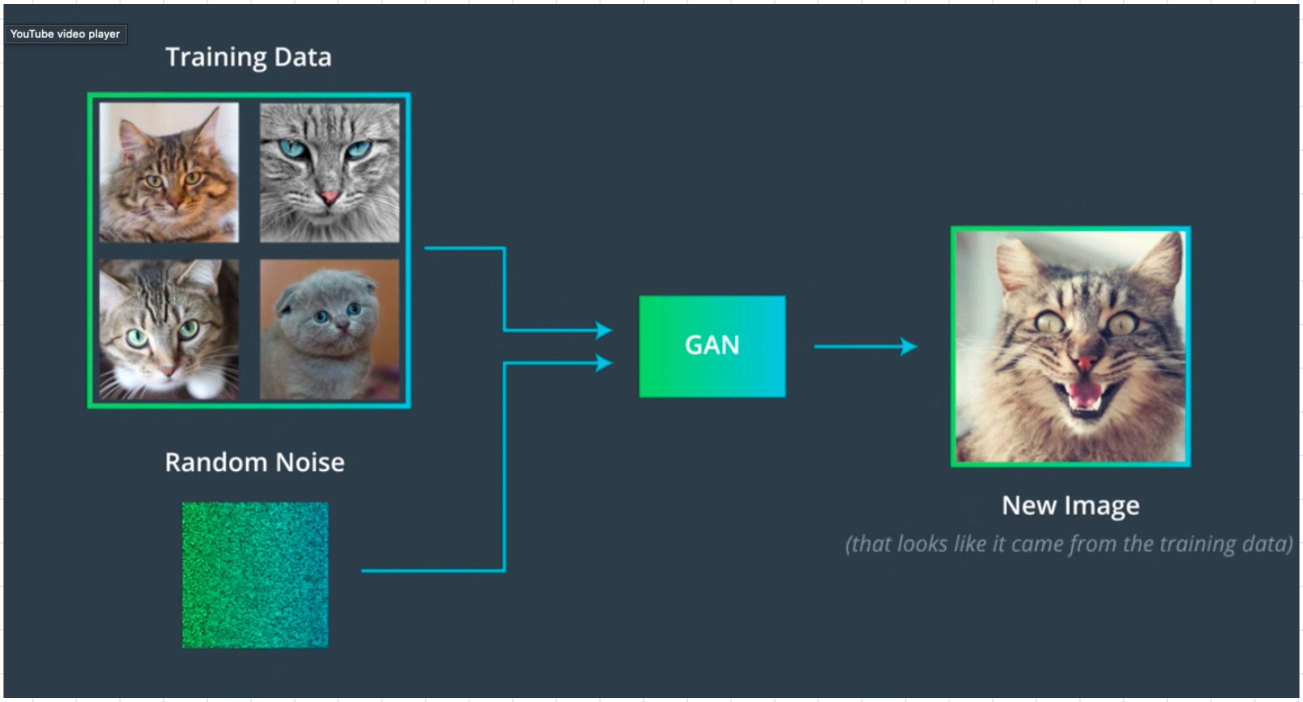
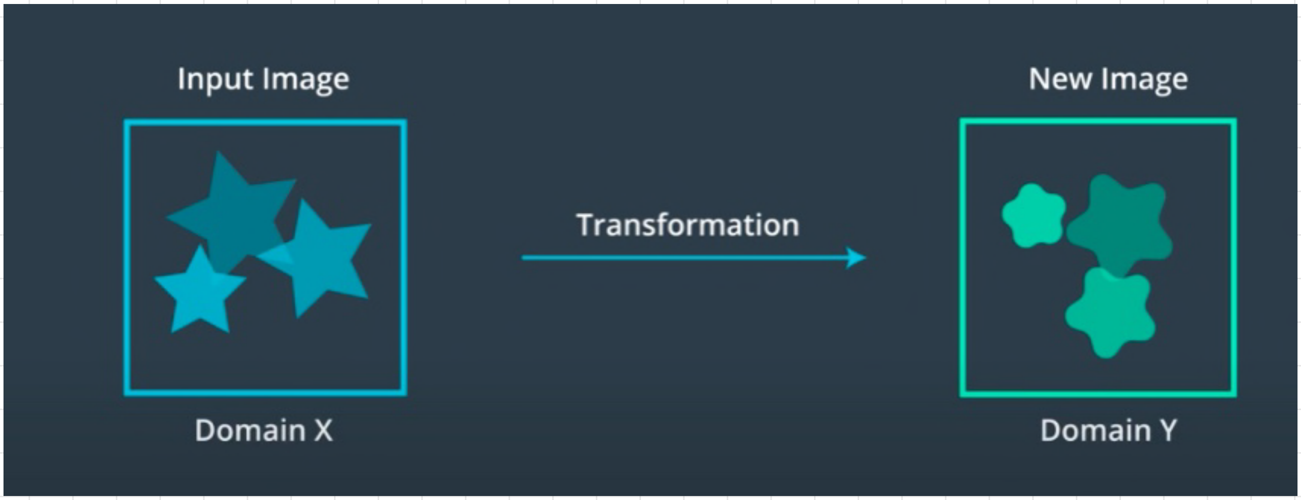
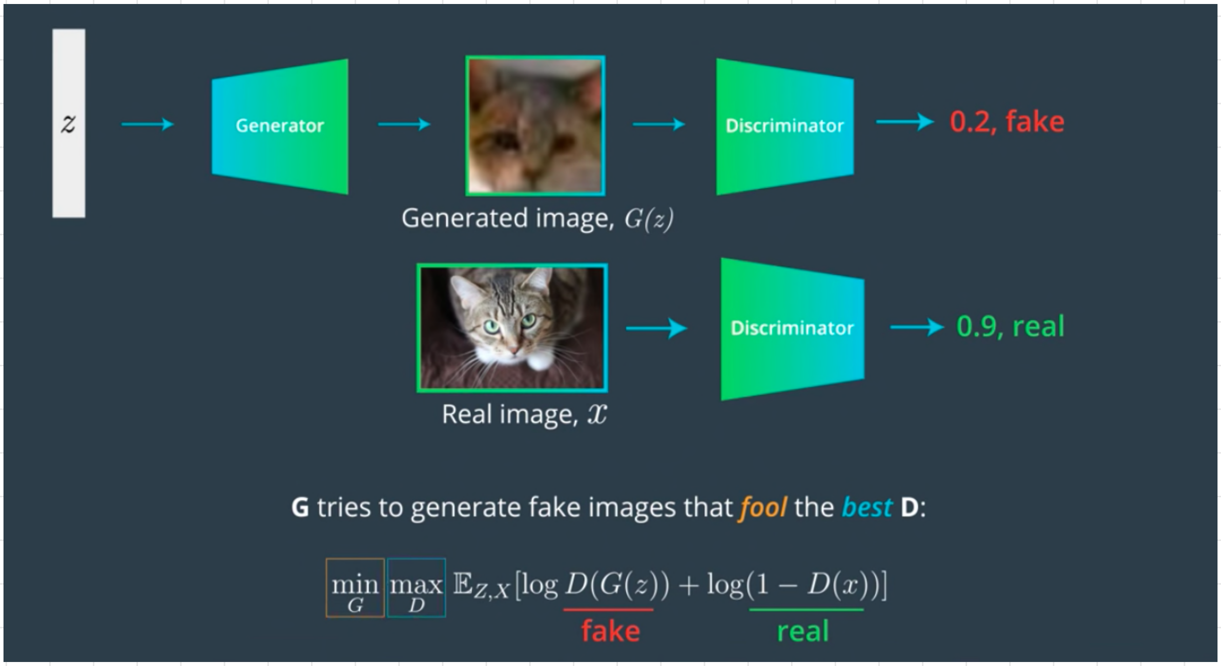


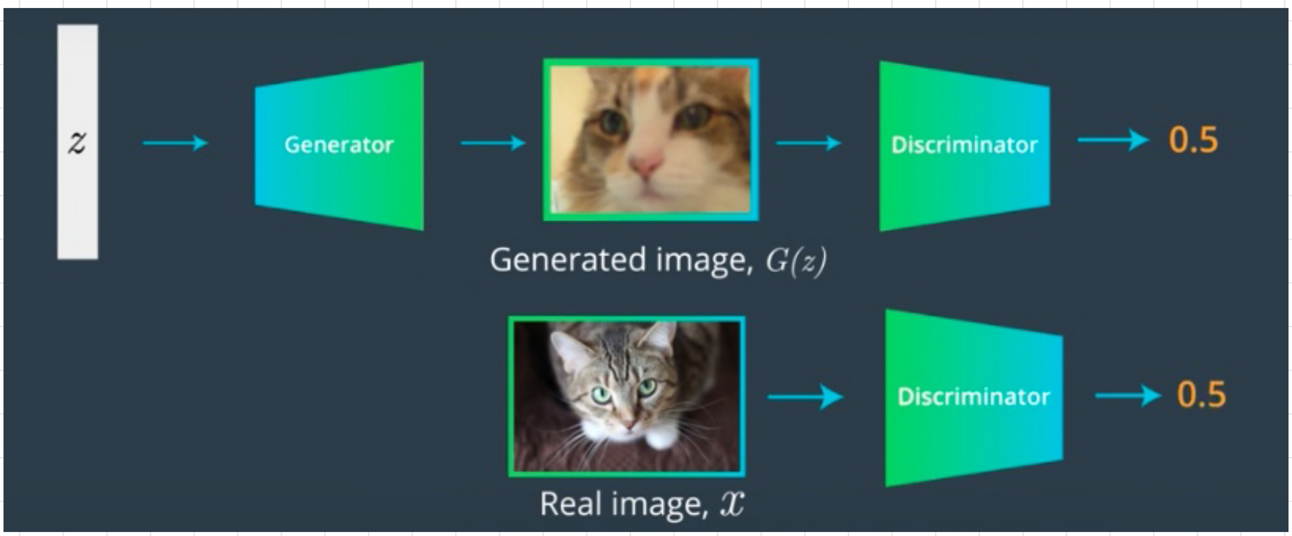
Image to Image Translation



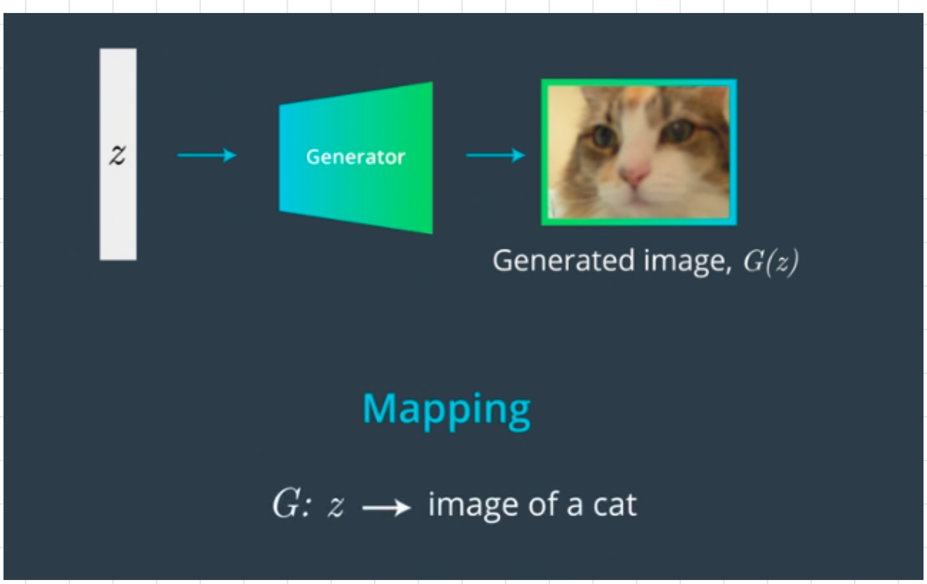


Designing Loss Function & Recap. GANs





Backprop until output $D(x) = \frac{1}{2}$ for any images



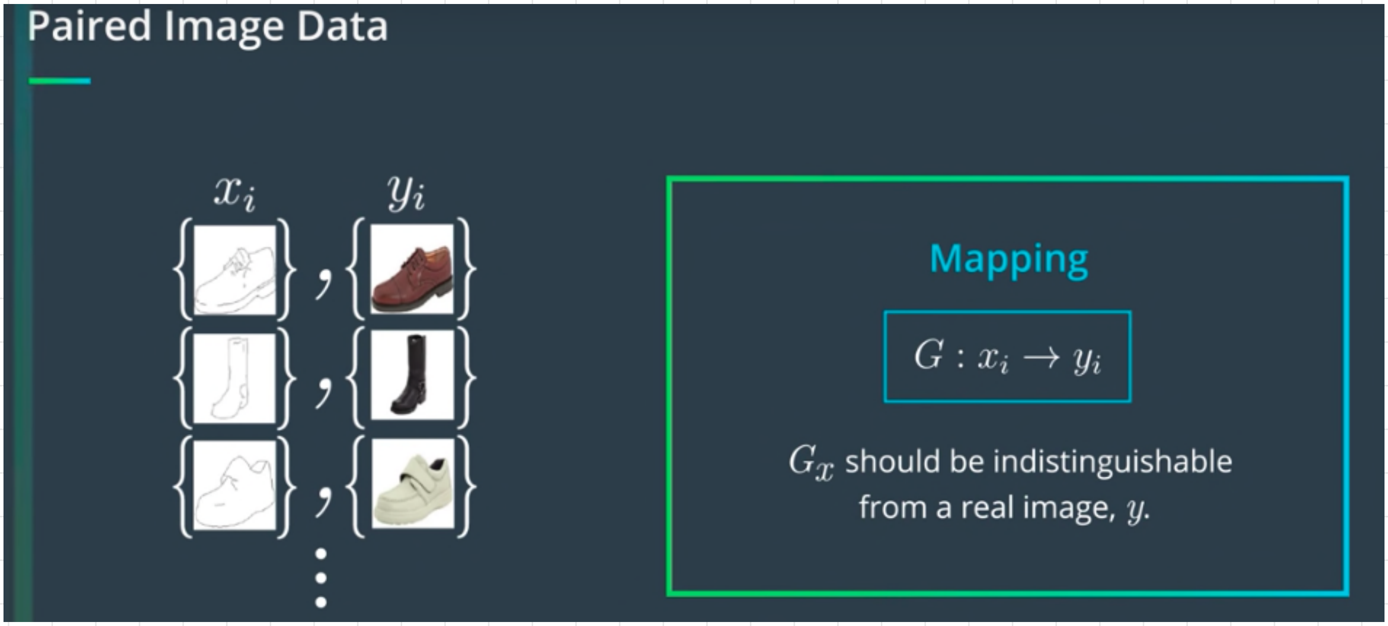
Basically G learns to map any random vectors z to a real img.

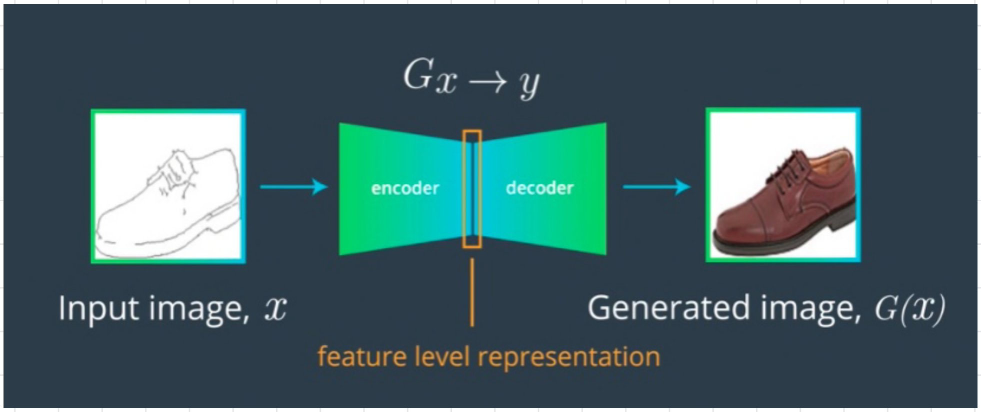
Take away:

- In GANs, the discriminator can be thought of as the loss function.
- We can devise a loss function, one that's learned, not explicitly found!

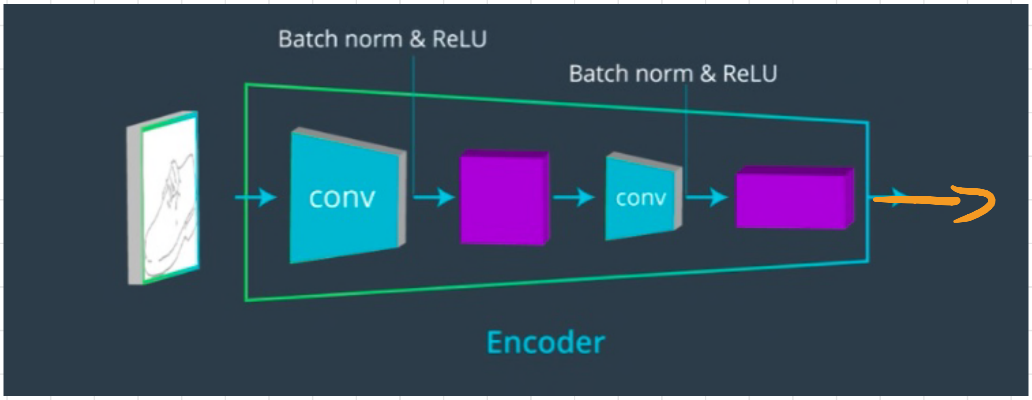
Pix2Pix

Generator



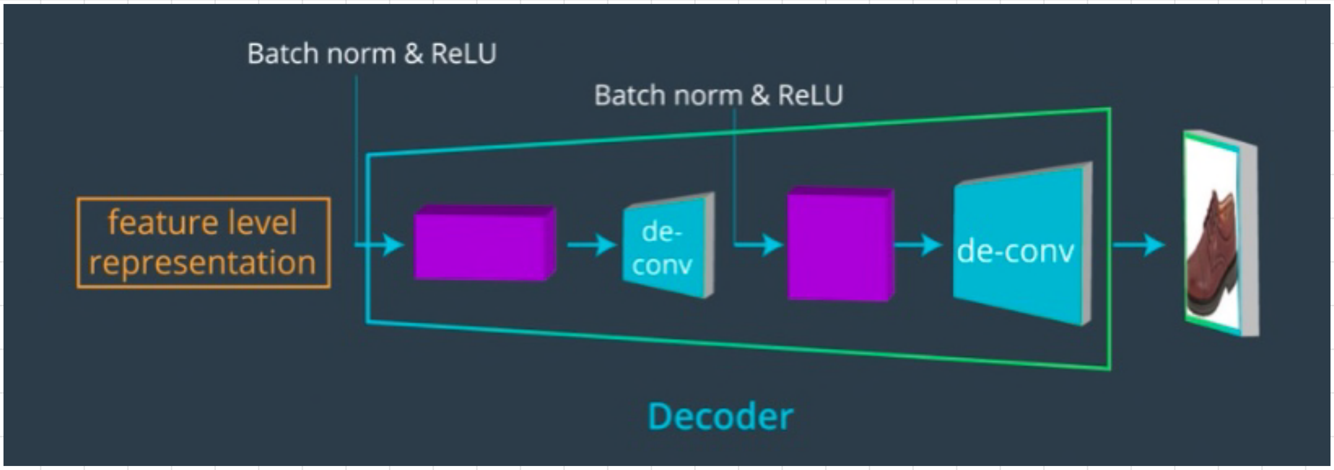


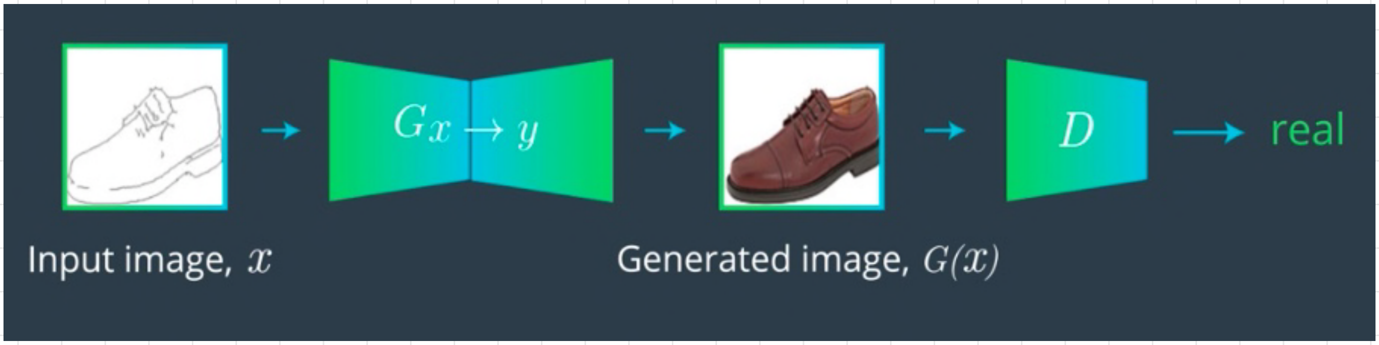
Instead of typical GANs that uses latent vectors z as input, we input an image.



Feature level Representation

Encoder distills spatial dims while preserving content.





Discriminator

Problem w/ D is that all it does is classifying real & fake images, regardless of their content.

Modified D:

Take in **pairs** of input & output imgs & output a value for **real pair** or **fake pair**.

How it works?

D takes in the input img & an **unknown img** by labeling an img as real or fake, it tries to determine whether or not the img is generated or not.

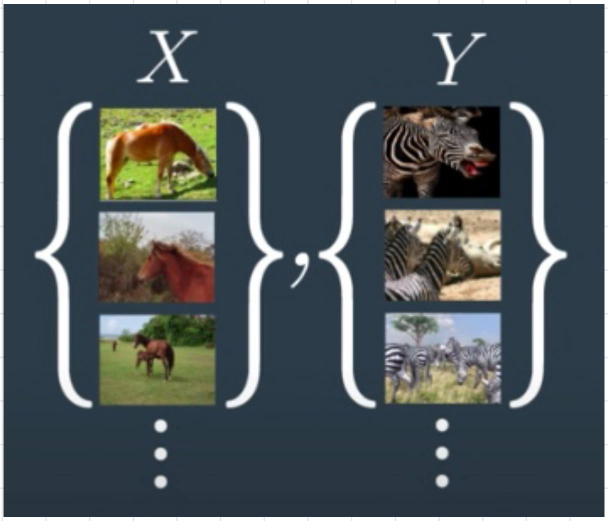
"Conditional Gain"

[Loss depends on both the input img & output img to the Generator]

Discriminator wants
- Real ← input to target pair
- False ← Generated & input pair

Generator wants the error to be **large!**

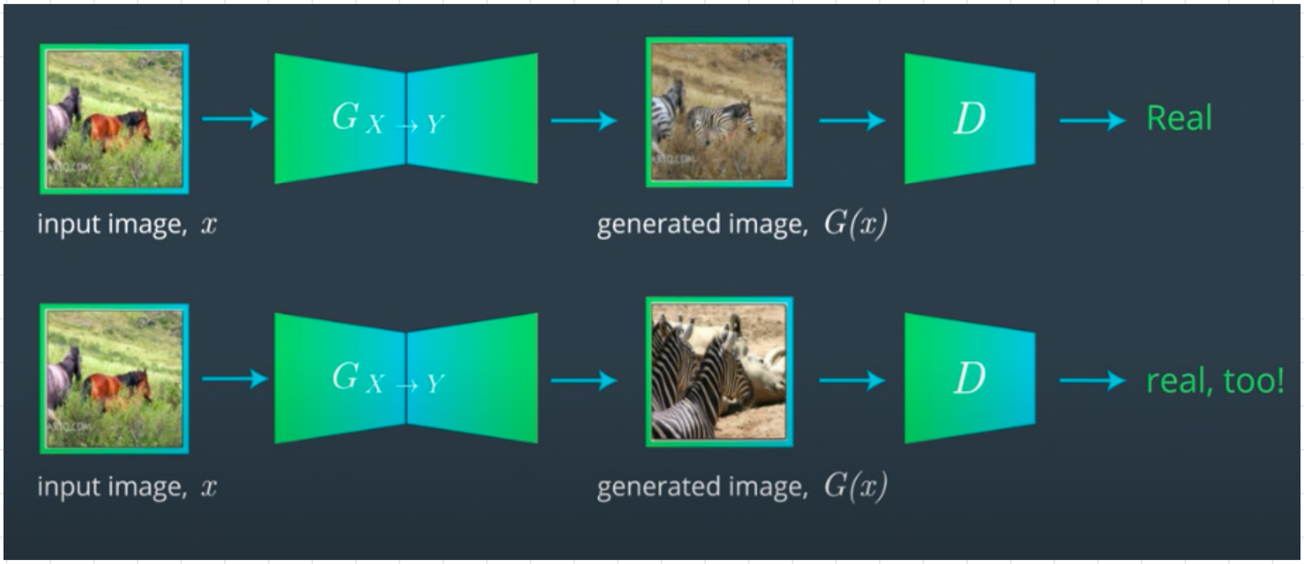
Cycle GANs

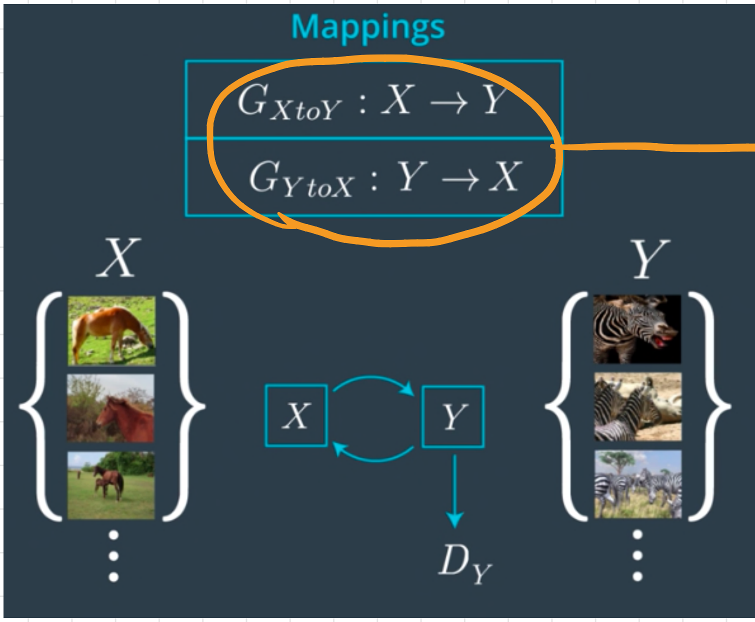


Mapping

$$G : X \to Y$$

- X indicates a set of images
- Lowercase x indicates a single image from set X.
- $G(x)$ should look like it comes from the set Y.





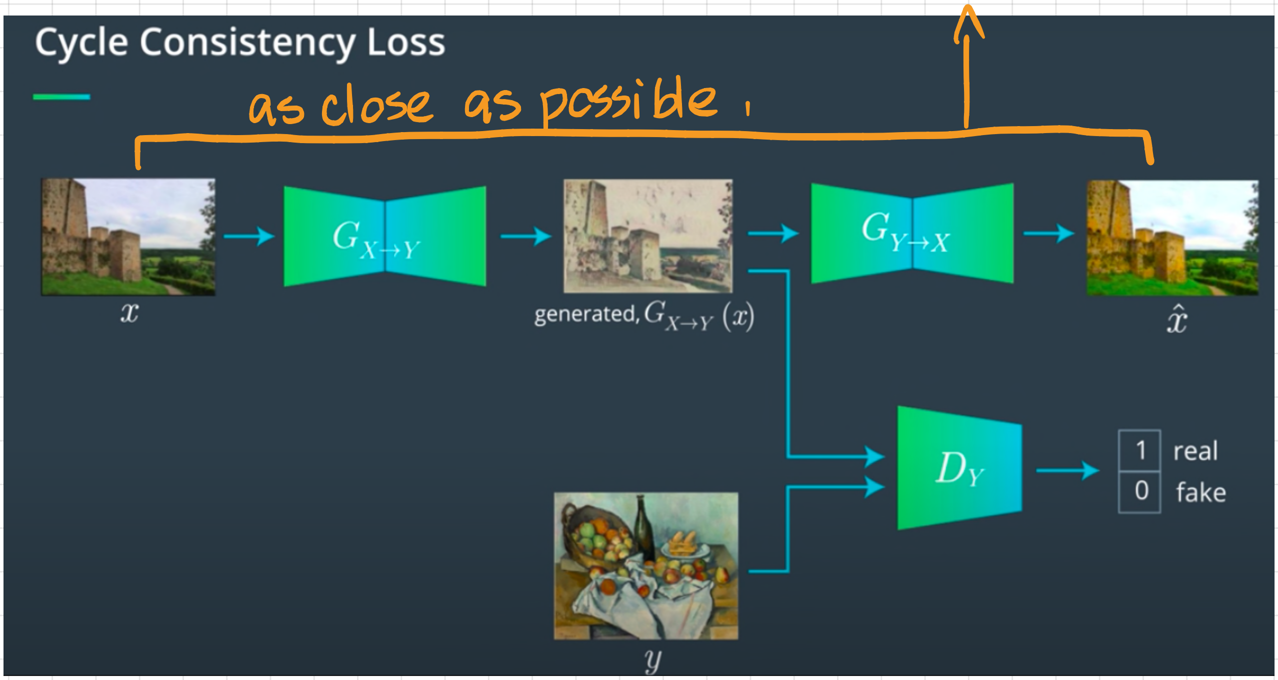
Cycle Consistency Constraint

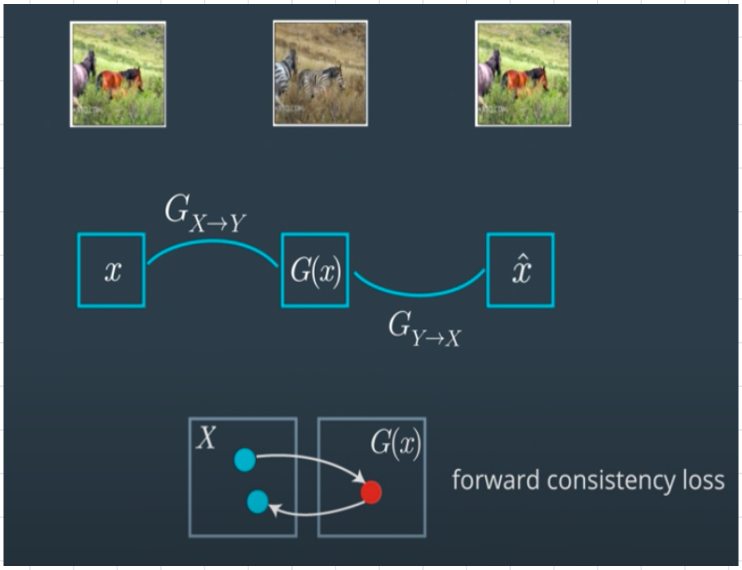
Additional mapping from Y to X.

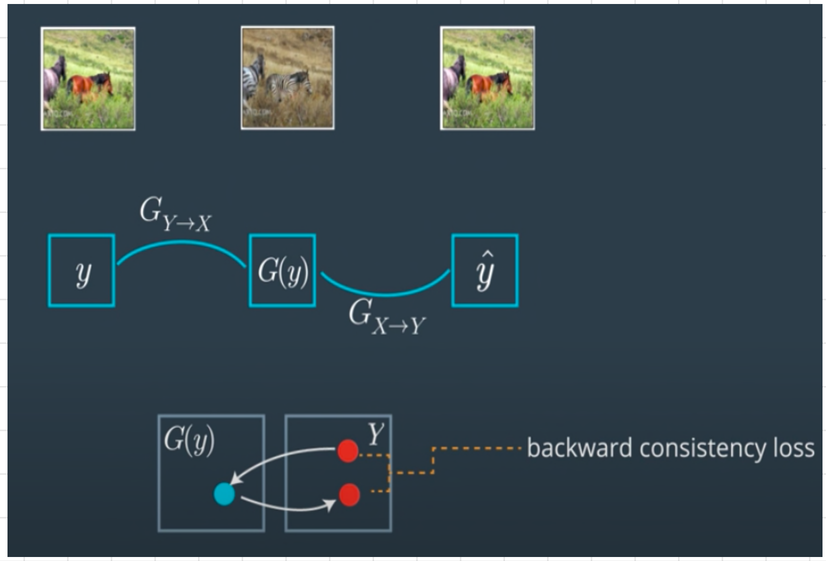
$$G_{Ytox}[G_{xtoY}(x)] \approx x$$

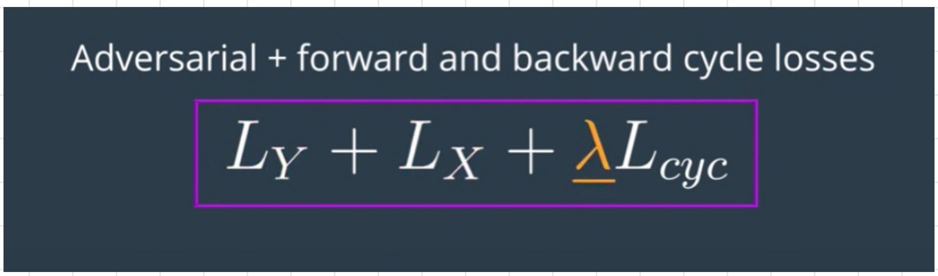
→ A Cycle GANs contains 2 adversarial discriminatory nets. D_x & D_y

Forward cycle consistency









Adversarial Loss:

Guide the network to produce images in a realistic x or y style

Why Does Cycle GANs Work?

We assume that the style & the content of the images are separated.

Encoder extracts features.

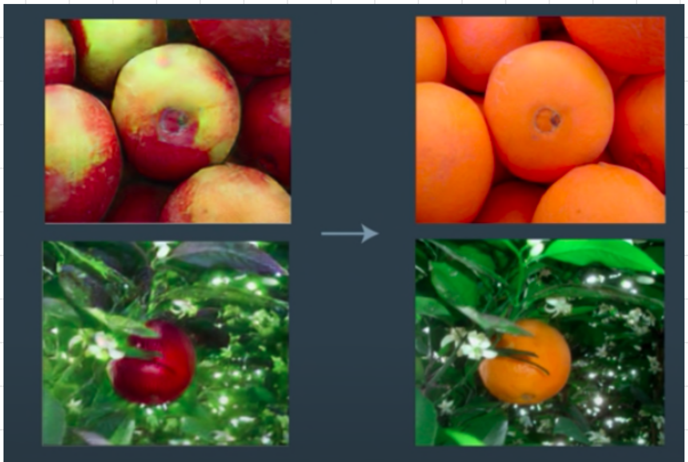
images $\longrightarrow$ feature maps

cares only about the ==content==

Generator keeps the content, producing another img of diff. style, reconstructing a good version of x.

Summary

- ==**Separation of Style & Content**==
- CycleGANs keep the **content** of the image & transfer the **style** of a whole collection x or y to it!

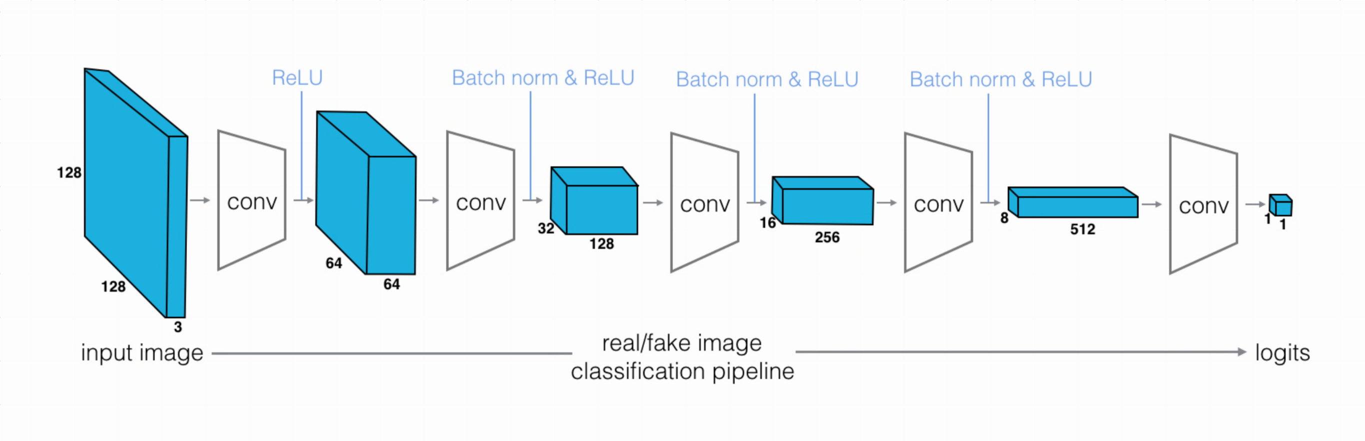


Apples to Oranges

Implementation

Discriminator

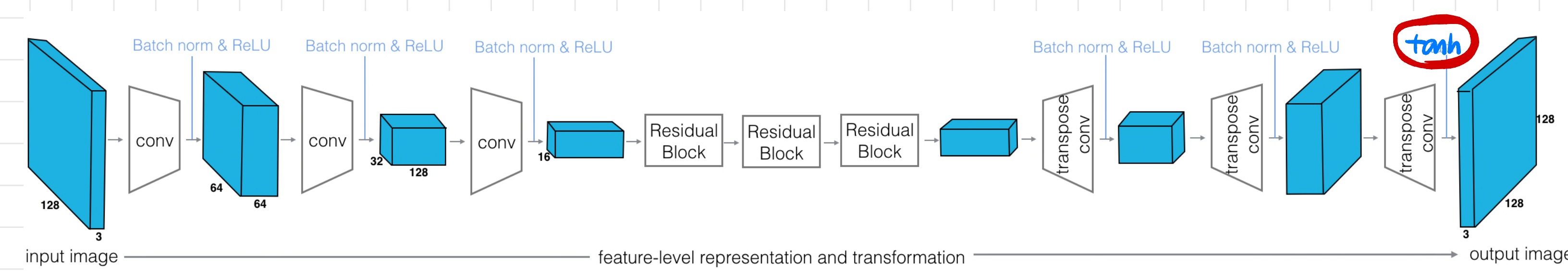
D_x, D_y see an image & classifies it as real or fake



No batch_norm at first or last layer.

Generator

Basically autoencoder w/ at least 6 res blocks
in between encoder & decoder



Residual Blocks

Connect the output of 1 layer & input of an earlier layer.

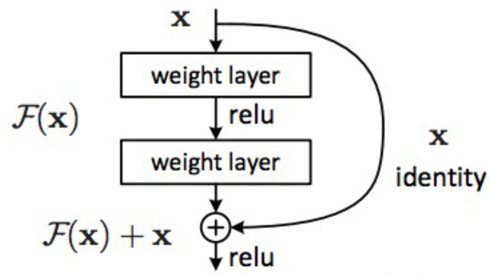


Figure 2. Residual learning: a building block.

Residual Function

- Usually when creating a net, the model learns a mapping M, from an input x to y.

$$M(x) = y$$

- Instead of ↑, we define a Res Function:

$$F(x) = M(x) - x$$

looking at the diff. between a mapping applied to x and the input x.

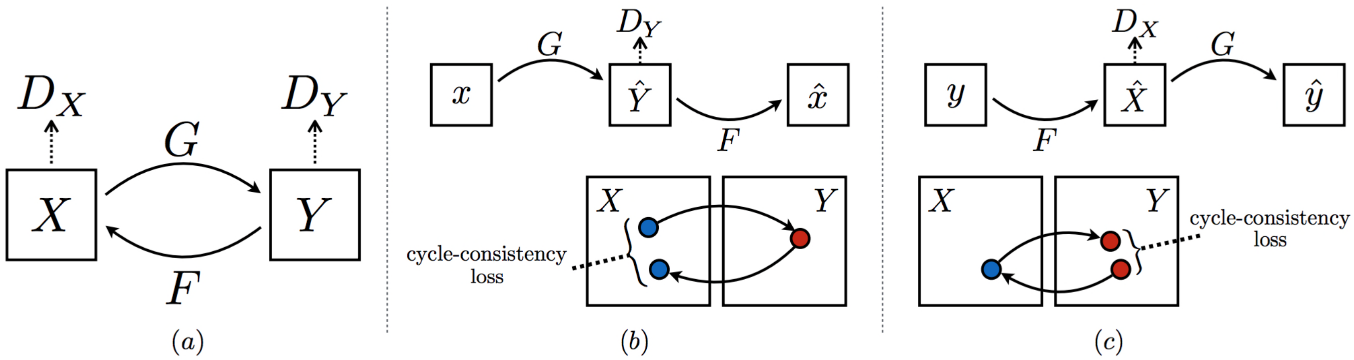
- F(x) is usually 2 conv + 1 norm layer + ReLU in bt.

have same # of inputs as outputs

$$M(x) = F(x) + x \longrightarrow y = F(x) + x$$

- Benefits:
  - Easier to optimize F(x) then M(x)
- If $F(x) = 0$, $M(x) = x$, $y = x$

Implementing Losses



Discriminator, real

- Discriminator takes in real imgs. (from X and Y) and check if they are real

D_out_Real_X : D_X (Real imgs_X) summer

D_out_Real_Y : D_Y (Real imgs_Y) winter

- Call in helper function to calculate losses

Real_mse_loss(D_out_Real_X)
Real_mse_loss(D_out_Real_Y)

Discriminator. Fake

- Generate imgs. X → Y   Y → X

Img_XtoY = G_XtoY (Image_X)

Img_YtoX = G_YtoX (Image_Y)

- Compute fake loss

For X to Y: D_Y(Img_ XtoY)

fake-mse-loss

Y to X: D_X(Img_YtoX)

...

Generator:

- Y to X: G_YtoX(Img_Y)

D_X(Img YtoX)

generate $\hat{Y}$ G_XtoY(...)

Cycle_consist. (Img Y, Y hat, λ)