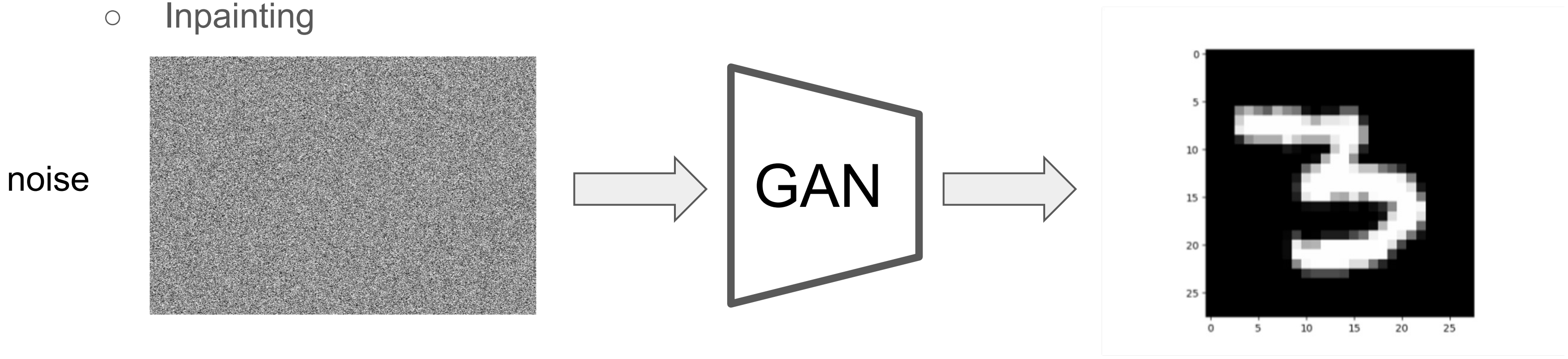
GAN: Generative Adversarial Network

Rowel Atienza
github.com/roatienza
University of the Philippines
2023

Why GANs?

- Used to generate high dim probability distributions
  - From one probability distribution to a target distribution
- Generation of realistic samples
  - Image super resolution
  - Art work synthesis
  - Image to image translation
  - Cross-domain transfer
  - Inpainting



Generative Models Landscape

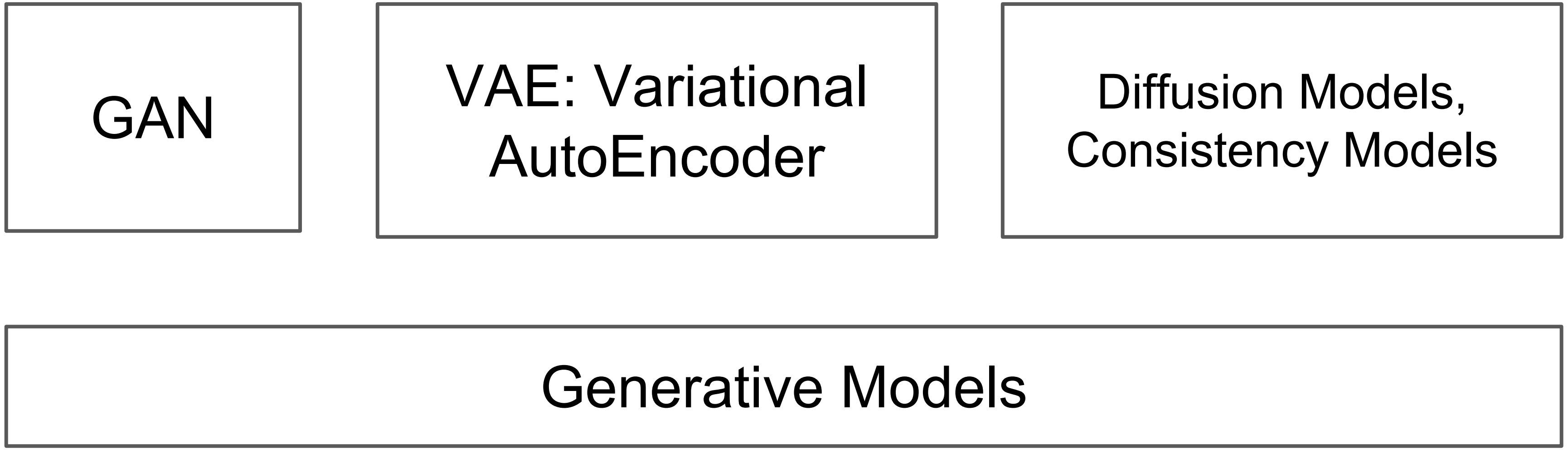

GAN
VAE: Variational AutoEncoder
Diffusion Models, Consistency Models
Generative Models

Image Super Resolution (SRGAN) [Ledig et al 2017]

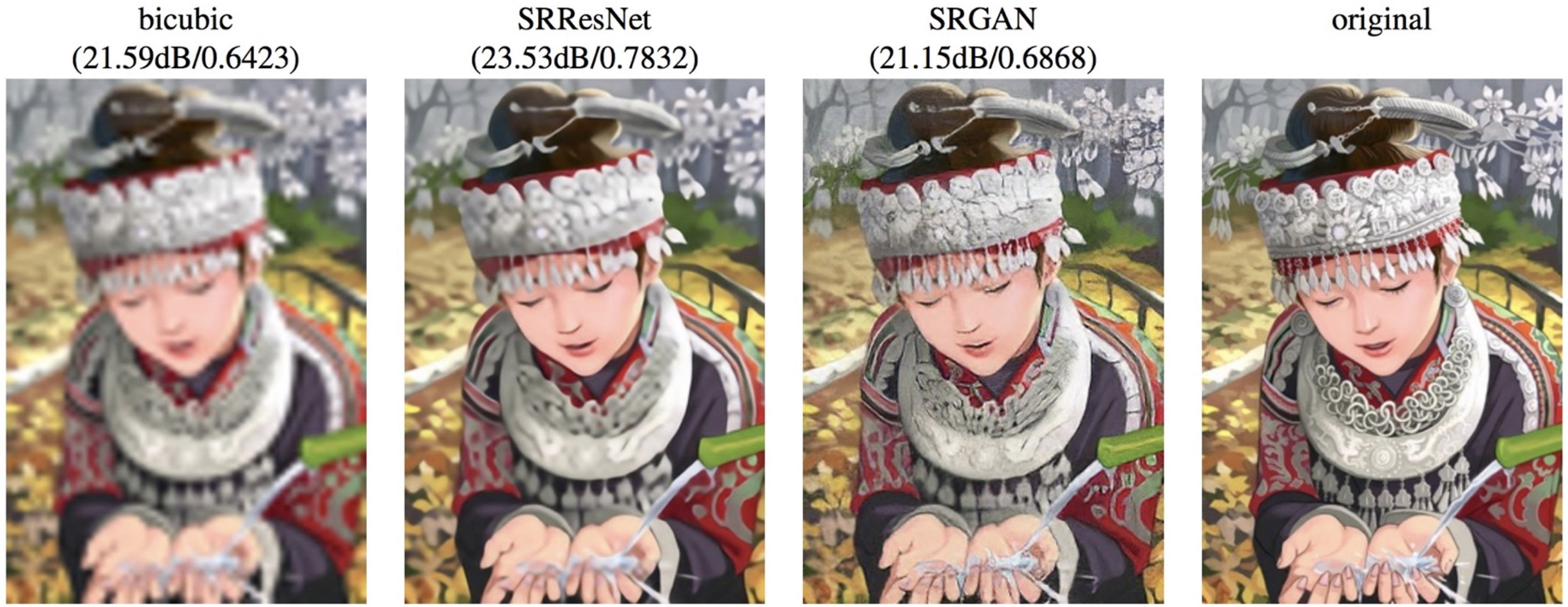
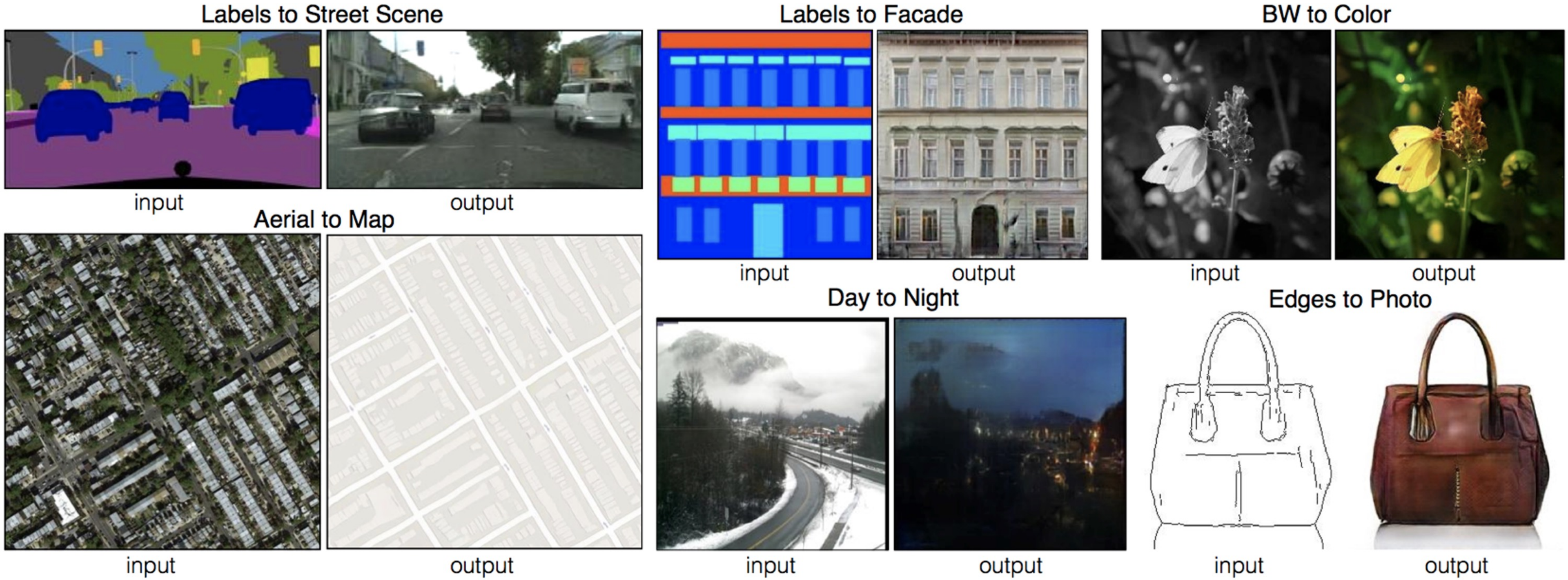


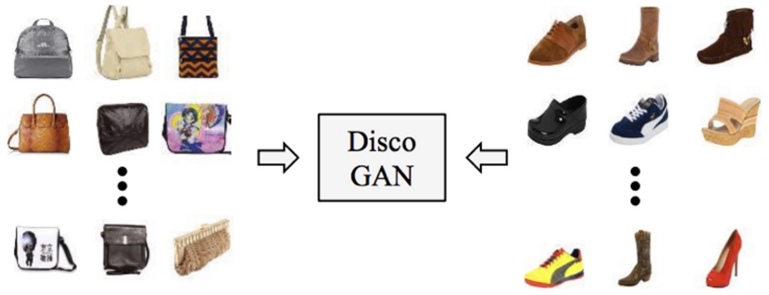
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

ProGANSR [Wang et al 2018]

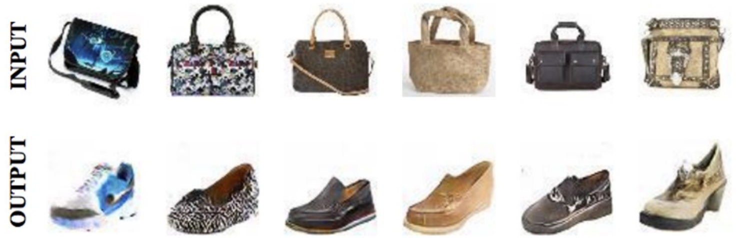
Image to Image Pix2Pix [Isola et al 2016]



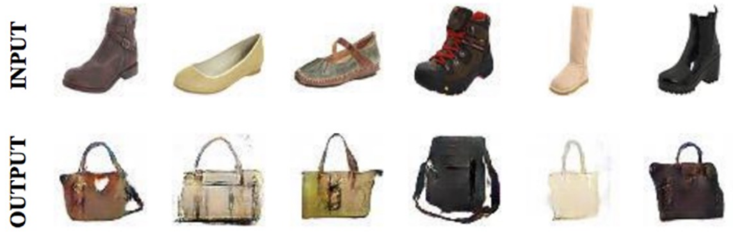
Cross-domain transfer (DiscoGAN) [Kim et al 2017]



(a) Learning cross-domain relations **without any extra label**



(b) Handbag images (input) & **Generated** shoe images (output)



(c) Shoe images (input) & **Generated** handbag images (output)

CycleGAN
Zhu et al
2017

Disclaimer: I was not part of this research project, I am merely providing commentary on this work.

Progressive Growing of GANs [Karras et al 2018]

StyleGAN [Karras et al 2019]

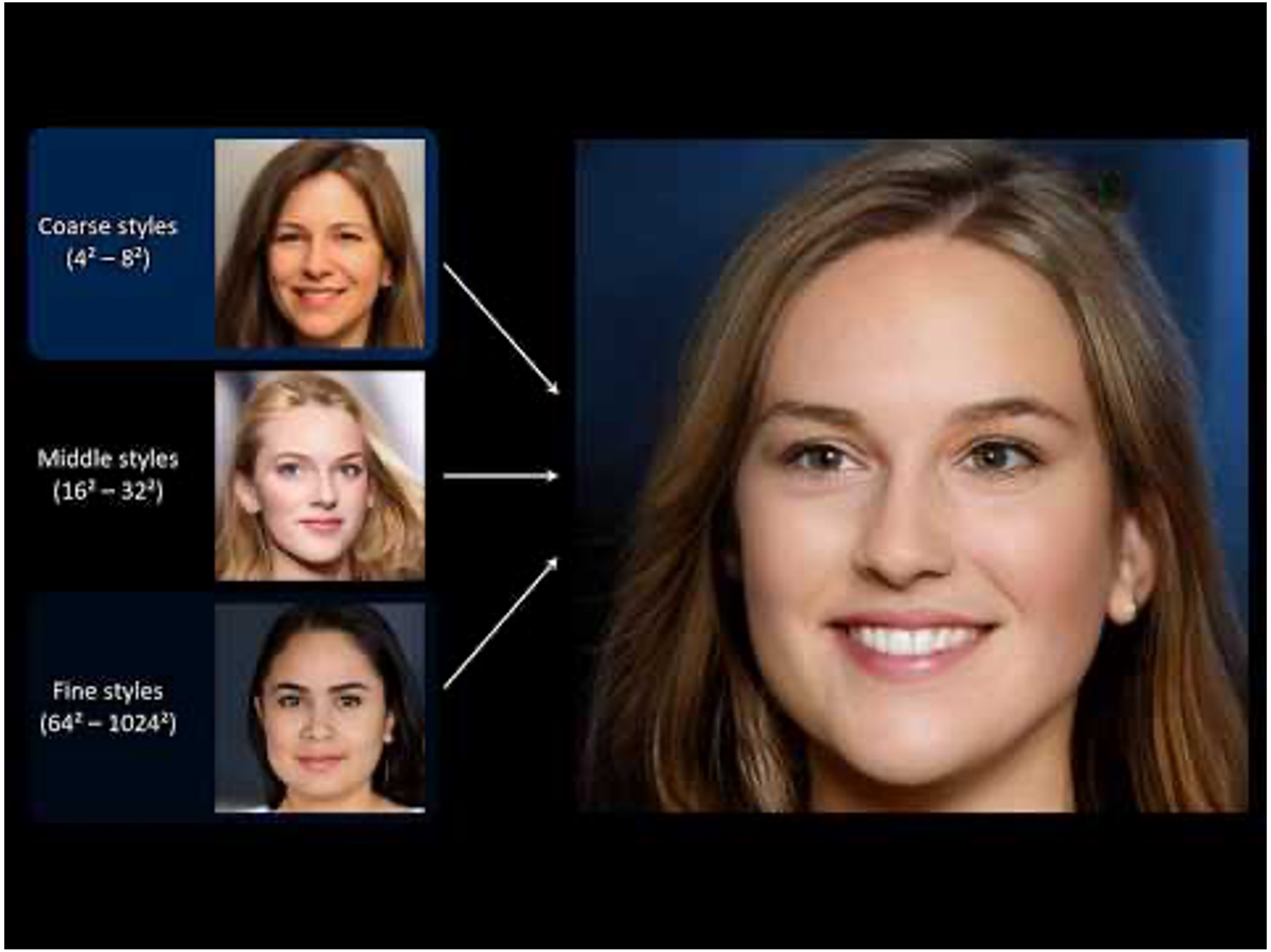
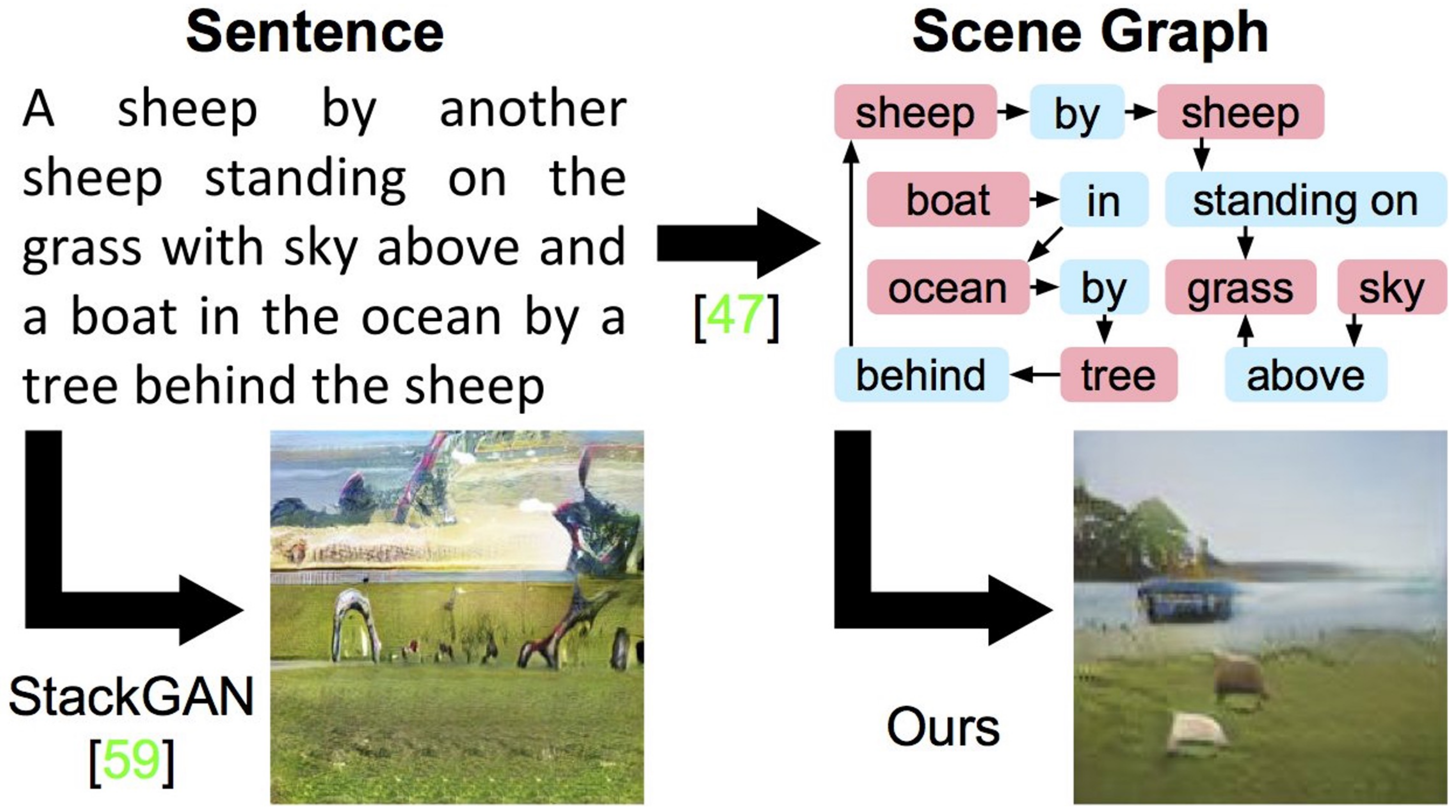


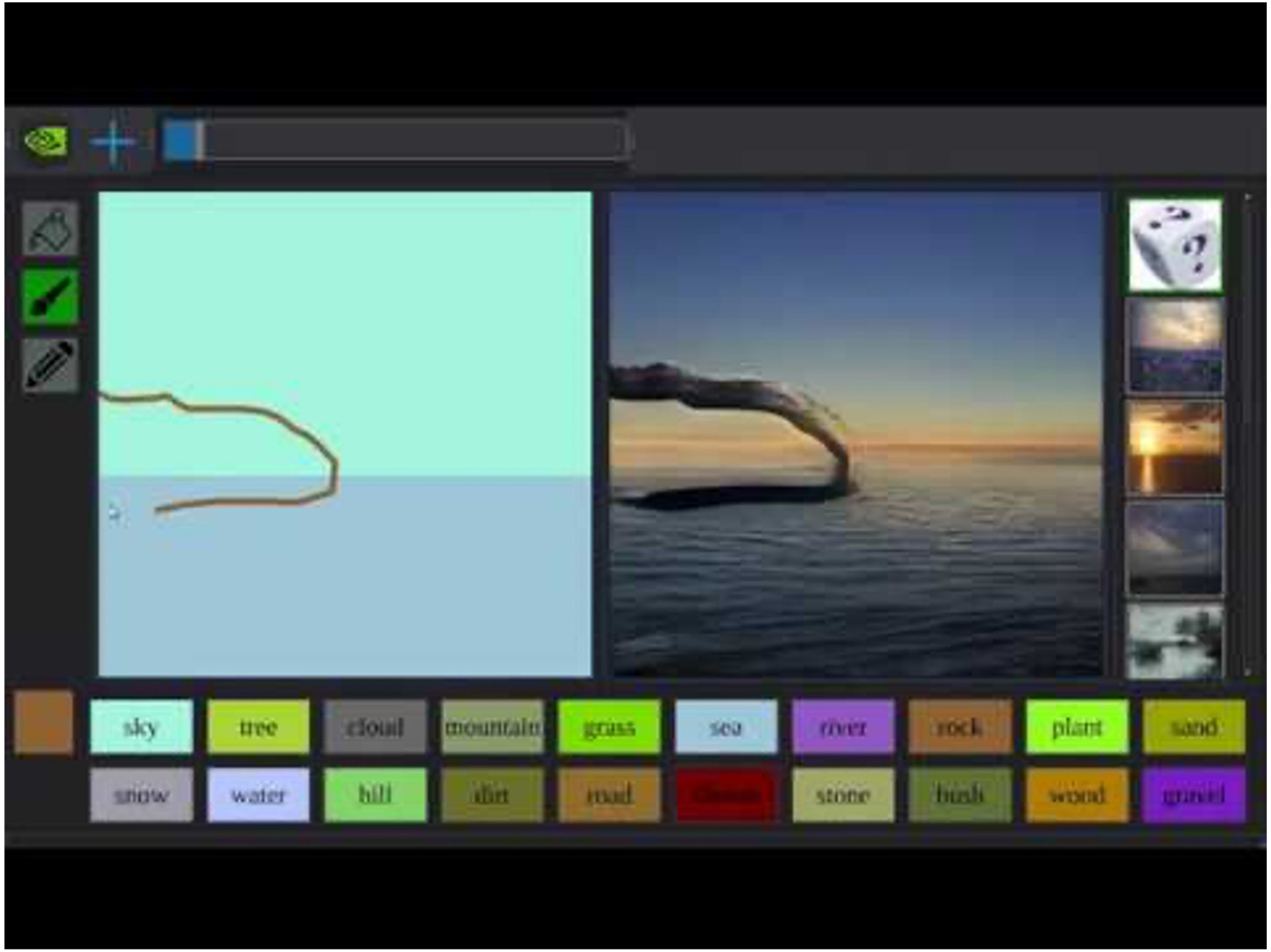
Image Generation [Johnson etal 2018]



A sheep by another sheep standing on the grass with sky above and a boat in the ocean by a tree behind the sheep

DeepFakes

DeepFakes


sky
tree
cloud
mountain
grass
sea
river
rock
plant
sand
snow
water
hill
dirt
road
stone
bush
wood
gravel

Maximum Likelihood Estimation (Basis of GAN)

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{model}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \underline{\log} \prod_{i=1}^{m} p_{model}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{\underline{i=1}}^{\boldsymbol{m}} \log p_{model}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

Find a model parameterized by $\boldsymbol{\theta}$ that estimates the input distribution

GAN - Maximum Likelihood Estimation

GAN aims to approximate the true data distribution from samples

MLE is minimizing KL Divergence

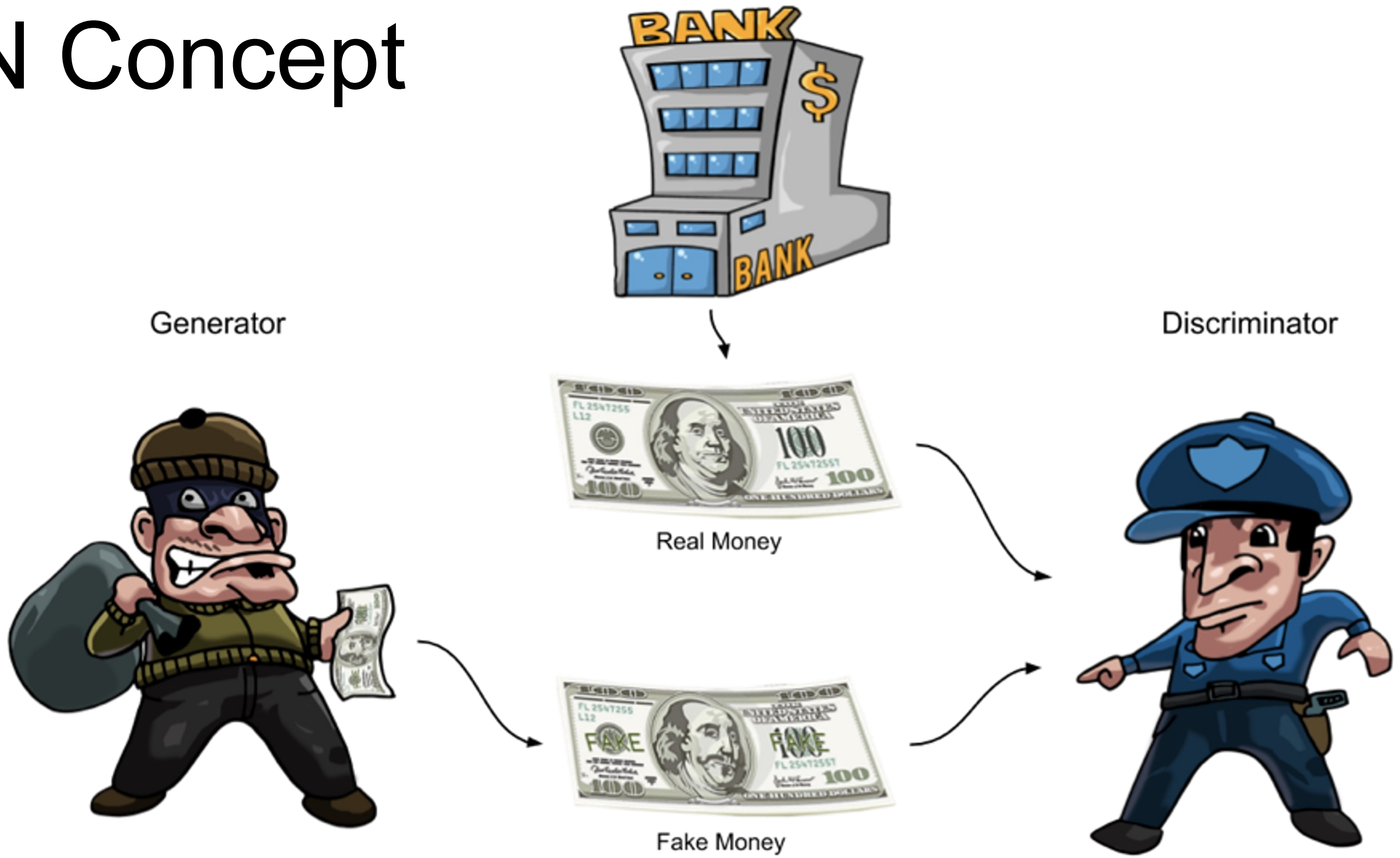
$$\boldsymbol{\theta}^* = arg\min_{\theta} D_{KL}(p_{data}(\boldsymbol{x}) \| p_{model}(\boldsymbol{x};\boldsymbol{\theta}))$$

- If $p_{data}(\boldsymbol{x})$ lies within $p_{model}(\boldsymbol{x};\boldsymbol{\theta})$, the model will recover $p_{data}(\boldsymbol{x})$
- In practice, we do have access to $p_{data}(\boldsymbol{x})$; only to training m samples from $p_{data}(\boldsymbol{x})$
- Using m samples, we approximate $p_{data}(\boldsymbol{x})$ by $\hat{p}_{data}(\boldsymbol{x})$ - an empirical distribution
- By minimizing the KL divergence between $\hat{p}_{data}(\boldsymbol{x})$ and $p_{model}(\boldsymbol{x};\boldsymbol{\theta})$ is equivalent to maximizing the log likelihood of training set

GAN Framework

- Two player system:
  - Generator and Discriminator
  - Both functions are differentiable wrt inputs and parameters
- Generator – creates samples that are intended to come from the training data
  - Fools the discriminator
- Discriminator – examines samples to determine whether they are fake (0) or real (1)
  - Trained using traditional supervised learning

GAN Concept



Counterfeiter prints fake money. It is labelled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labelling the fake money as real.

The police are trained to spot real from fake money. Sometimes, the police give feedback to the counterfeiter why the money is fake.

GAN Framework

Discriminator
$$\begin{tikzcd} \boldsymbol{x} \arrow[r, "\boldsymbol{\theta}^{(D)}"] & \boldsymbol{y} \end{tikzcd}$$
$$\mathcal{L}^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

Trainable from D

Generator
$$\begin{tikzcd} \boldsymbol{z} \arrow[r, "\boldsymbol{\theta}^{(G)}"] & \boldsymbol{x} \end{tikzcd}$$
$$\mathcal{L}^{(G)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

Trainable from G

Loss Functions

Discriminator:

$$\mathcal{L}^{(D)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log \mathcal{D}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}} \log\left(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{z}))\right)$$

Generator:

$$\mathcal{L}^{(G)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{z}} \log \mathcal{D}(\mathcal{G}(\boldsymbol{z}))$$

Loss Functions - In English

Discriminator:

$$\mathcal{L}^{(D)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log \mathcal{D}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}} \log\left(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{z}))\right)$$

Learn to identify real (1) from fake (0)

Generator:

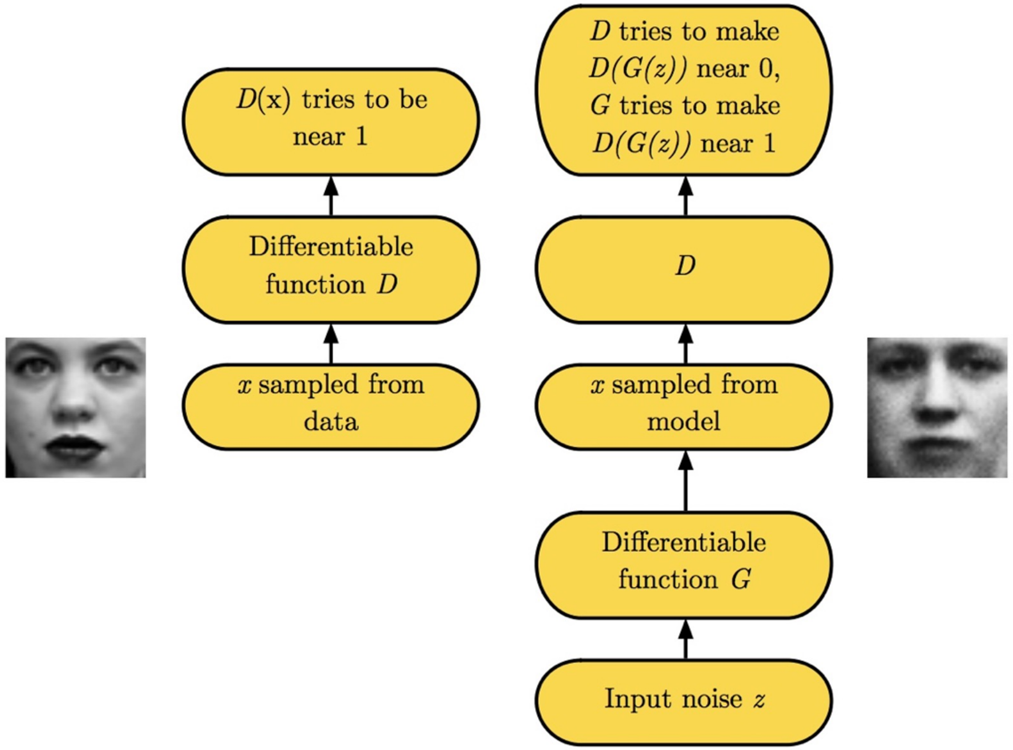
$$\mathcal{L}^{(G)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{z}} \log \mathcal{D}(\mathcal{G}(\boldsymbol{z}))$$

Learn to fool the discriminator by pretending to be real (1)

GAN Training

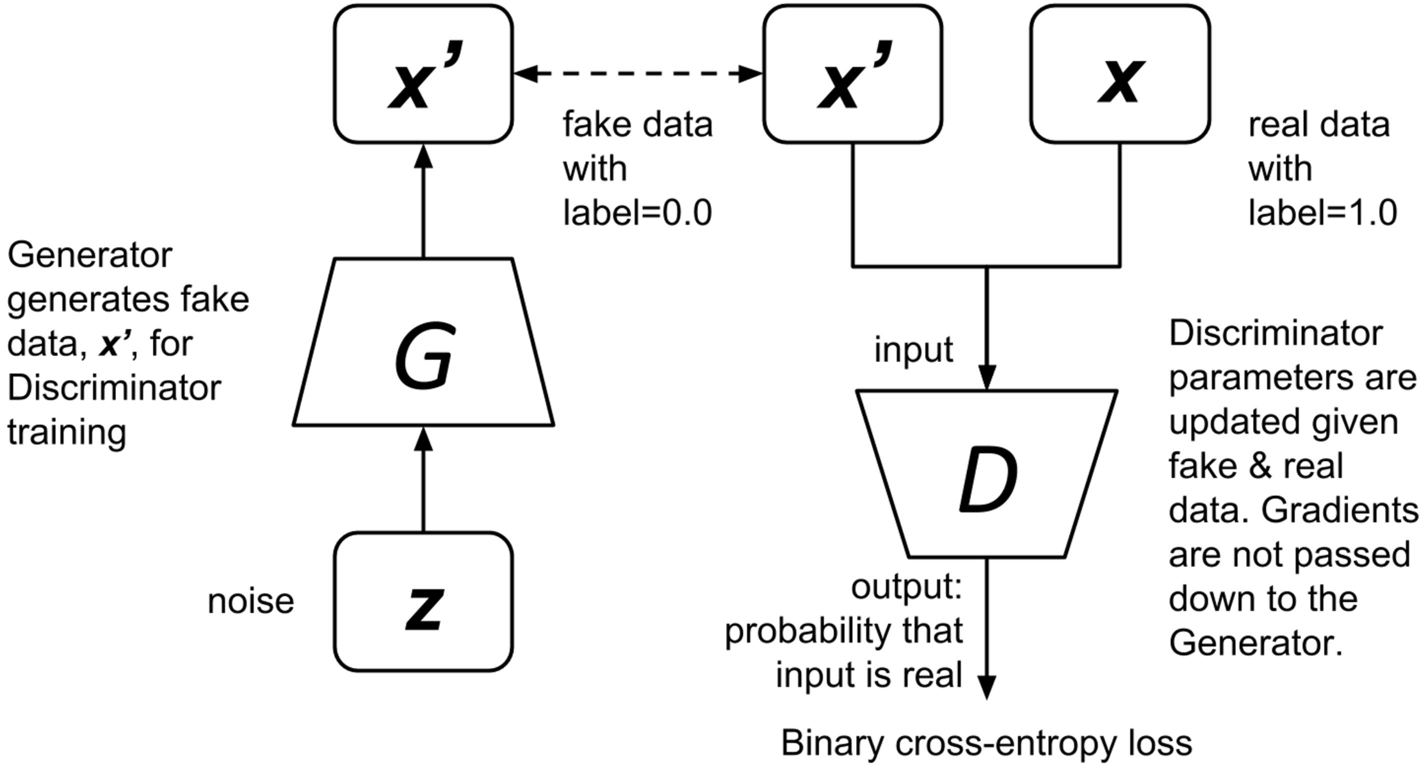
1. Unfreeze Discriminator parameters
2. Discriminator training - minimize $\mathcal{L}^{(D)}$ through $\boldsymbol{\theta}^{(D)}$
   - Minibatch from real $\boldsymbol{x}$ labelled real (1)
   - Minibatch from fake $\boldsymbol{x} = G(\boldsymbol{z})$ labelled as fake (0)
3. Freeze Discriminator parameters
4. Generator training through Adversarial training - minimize $\mathcal{L}^{(G)}$ through $\boldsymbol{\theta}^{(G)}$
   - Minibatch from fake $\boldsymbol{x} = G(\boldsymbol{z})$ pretending to be real (1)
5. Repeat 1 – 4 until loss conditions are met

GAN Training [Goodfellow 2016]



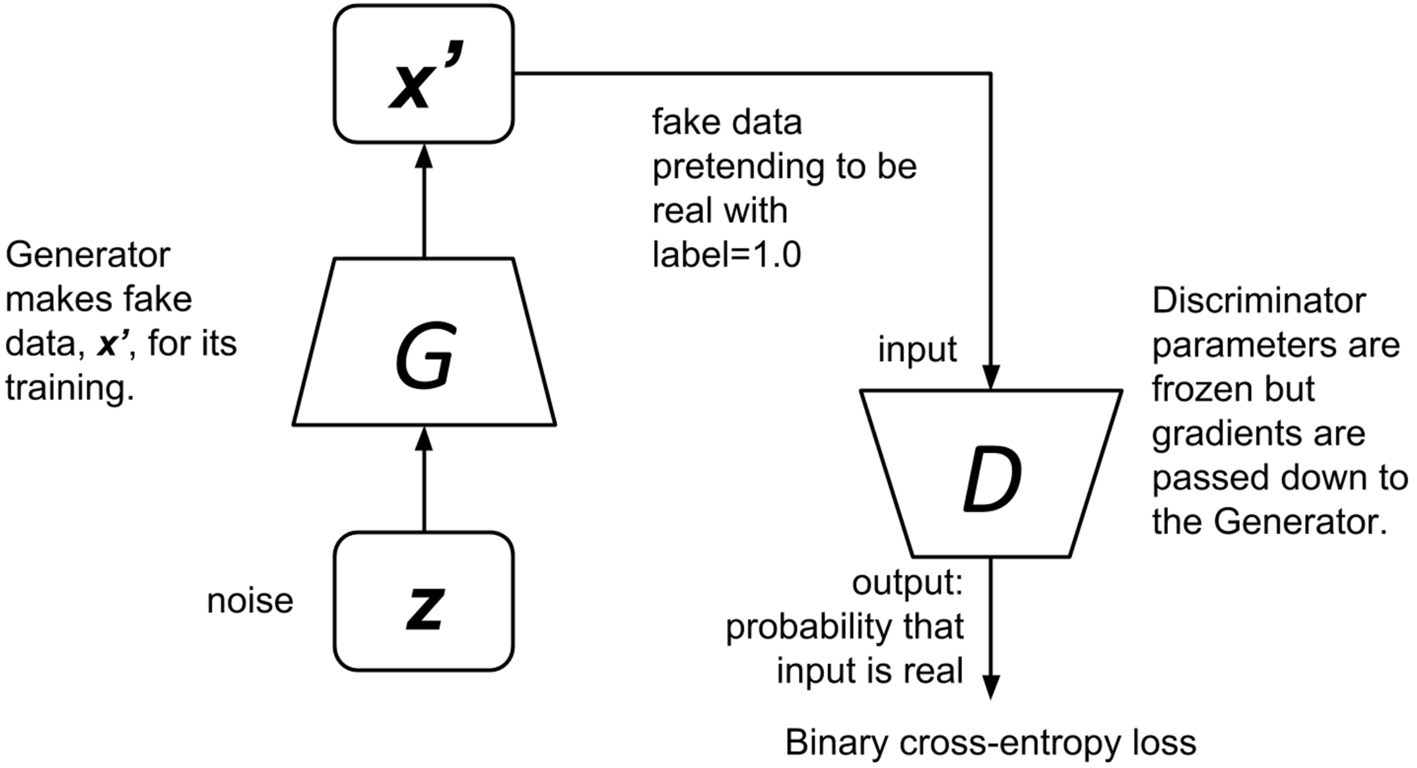
D(x) tries to be near 1
Differentiable function D
x sampled from data
D tries to make $D(G(z))$ near 0, G tries to make $D(G(z))$ near 1
D
x sampled from model
Differentiable function G
Input noise z

DCGAN Discriminator Training



x'
x'
x
fake data with label=0.0
real data with label=1.0
Generator generates fake data, x', for Discriminator training
G
input
Discriminator parameters are updated given fake & real data. Gradients are not passed down to the Generator.
D
noise
z
output: probability that input is real
Binary cross-entropy loss

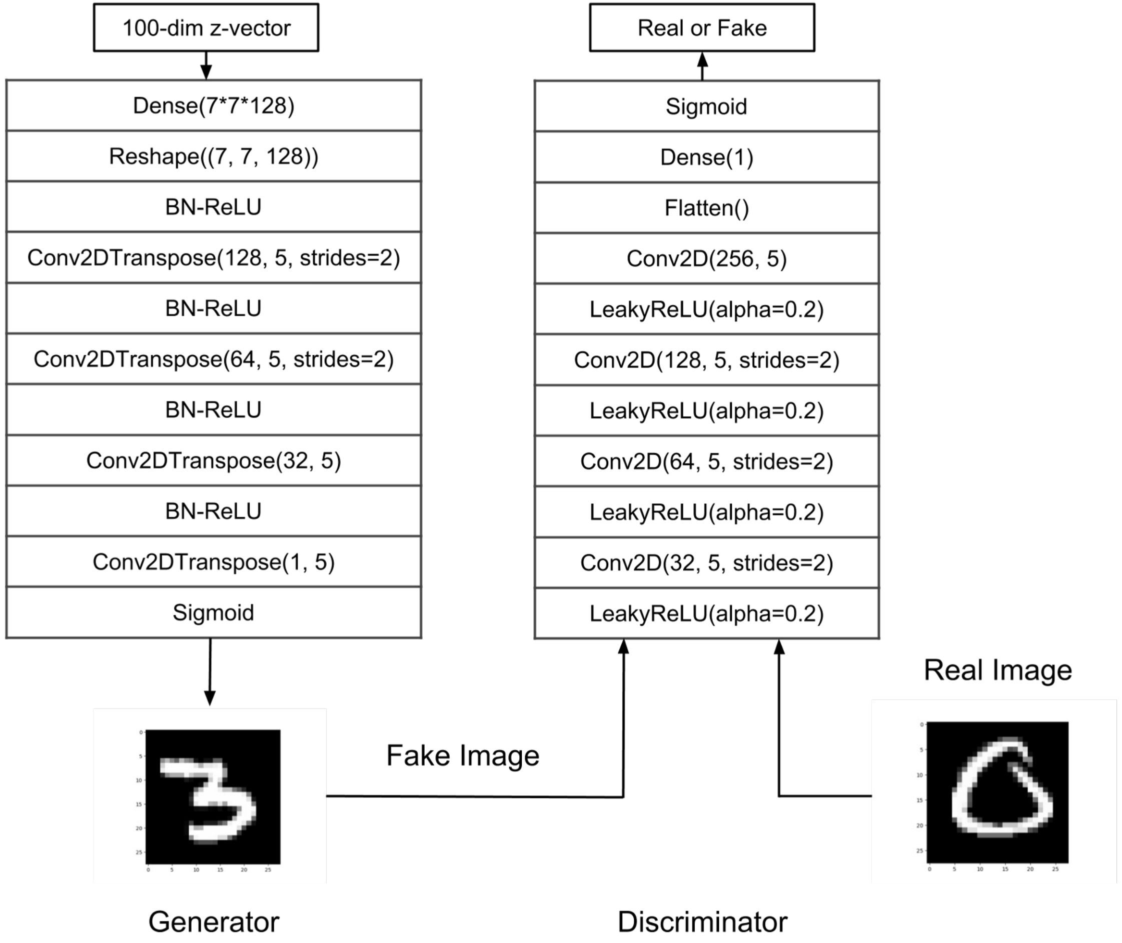
DCGAN Generator Training



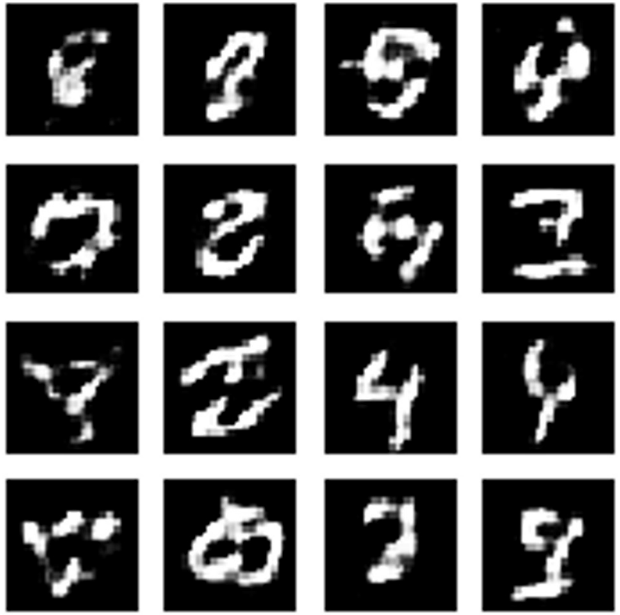
x'
fake data pretending to be real with label=1.0
Generator makes fake data, ***x'***, for its training.
G
noise
z
input
D
Discriminator parameters are frozen but gradients are passed down to the Generator.
output: probability that input is real
Binary cross-entropy loss

DCGAN [Radford 2016] in Keras

DCGAN is a GAN implementation using Deep CNN



DCGAN Outputs



Conditional GAN [Mirza and Osindero 2014]

- GAN has no control on which digits to generate from noise
- CGAN - a variation of GAN that imposes a constraint (eg which digit) to control the attribute of the generator output

CGAN Loss Functions

Discriminator:

$$\mathcal{L}^{(D)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log \mathcal{D}(\boldsymbol{x}|\boldsymbol{y}) - \mathbb{E}_{\boldsymbol{z}} \log\left(1 - \mathcal{D}\left(\mathcal{G}(\boldsymbol{z}|\boldsymbol{y})\right)\right)$$

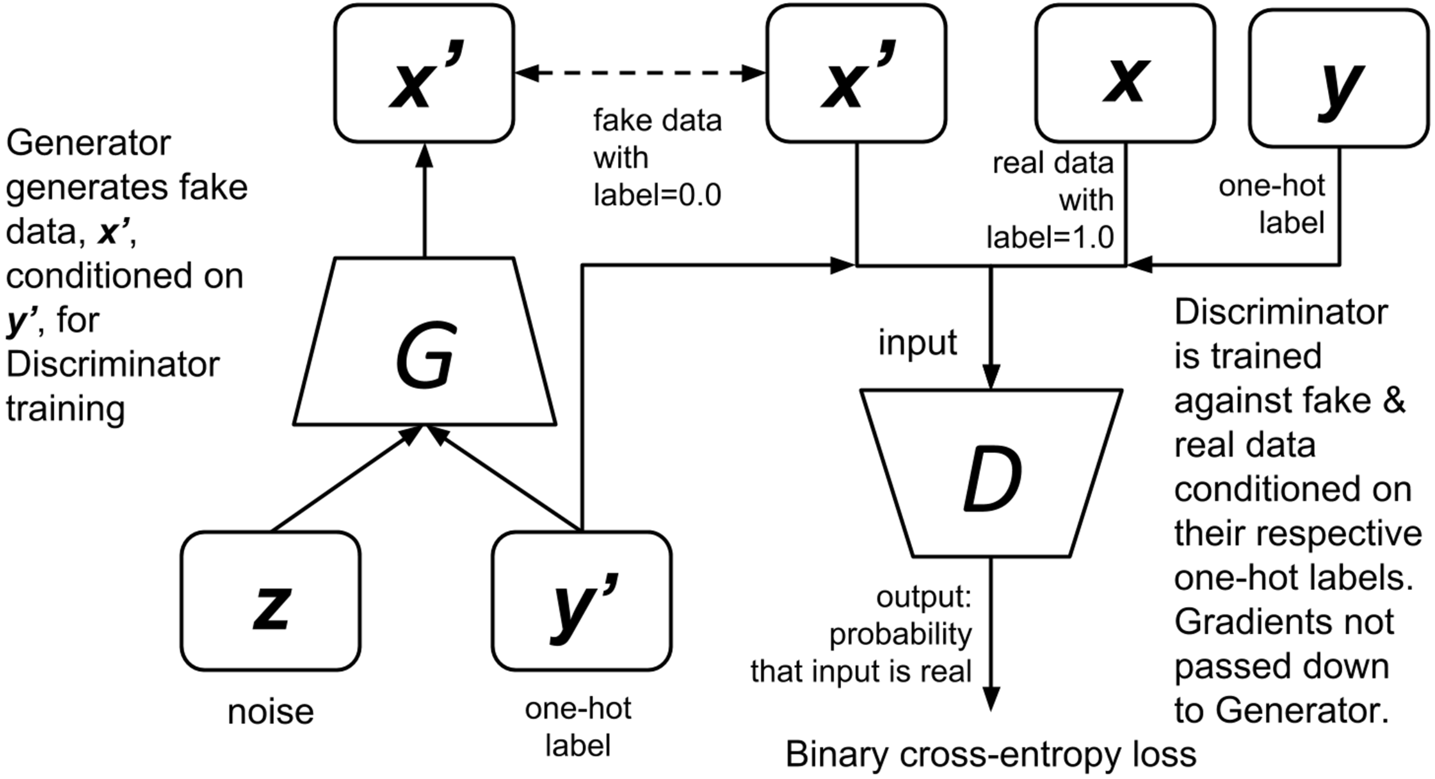
Learn to identify real (1) given condition ***y*** from fake (0) given condition ***y***

Generator:

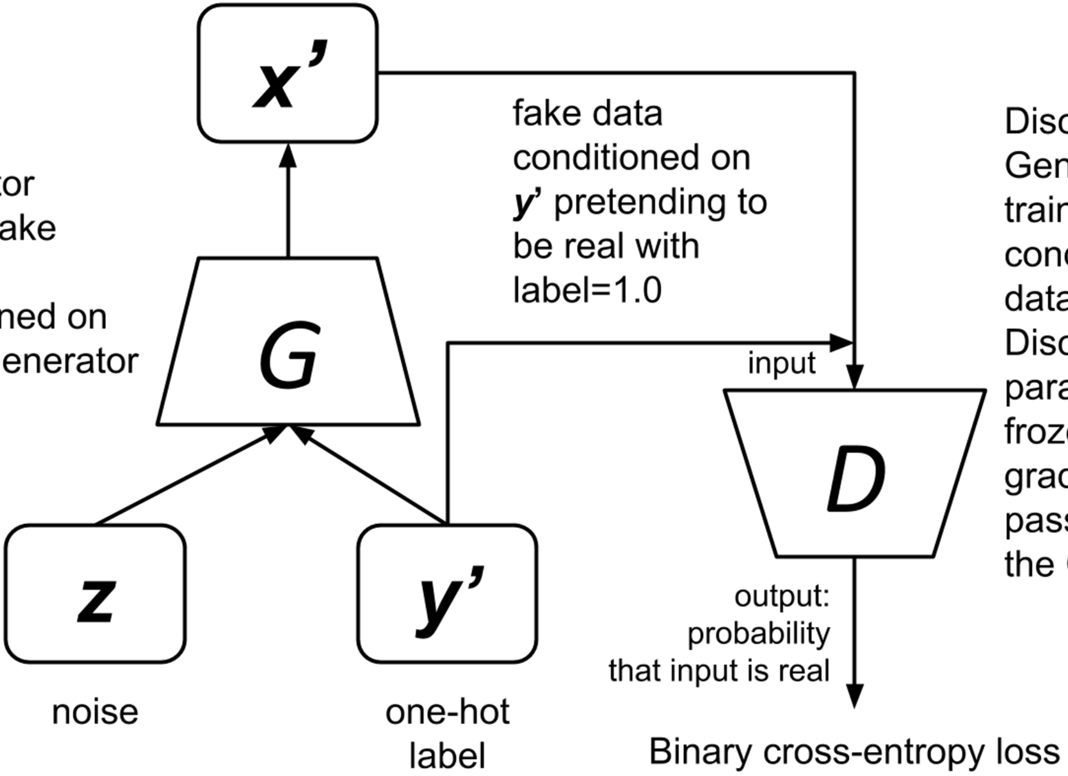
$$\mathcal{L}^{(G)}\left(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}\right) = -\mathbb{E}_{\boldsymbol{z}} \log \mathcal{D}\left(\mathcal{G}(\boldsymbol{z}|\boldsymbol{y})\right)$$

Learn to fool the discriminator by pretending to be real (1) given condition ***y***

CGAN Discriminator Training

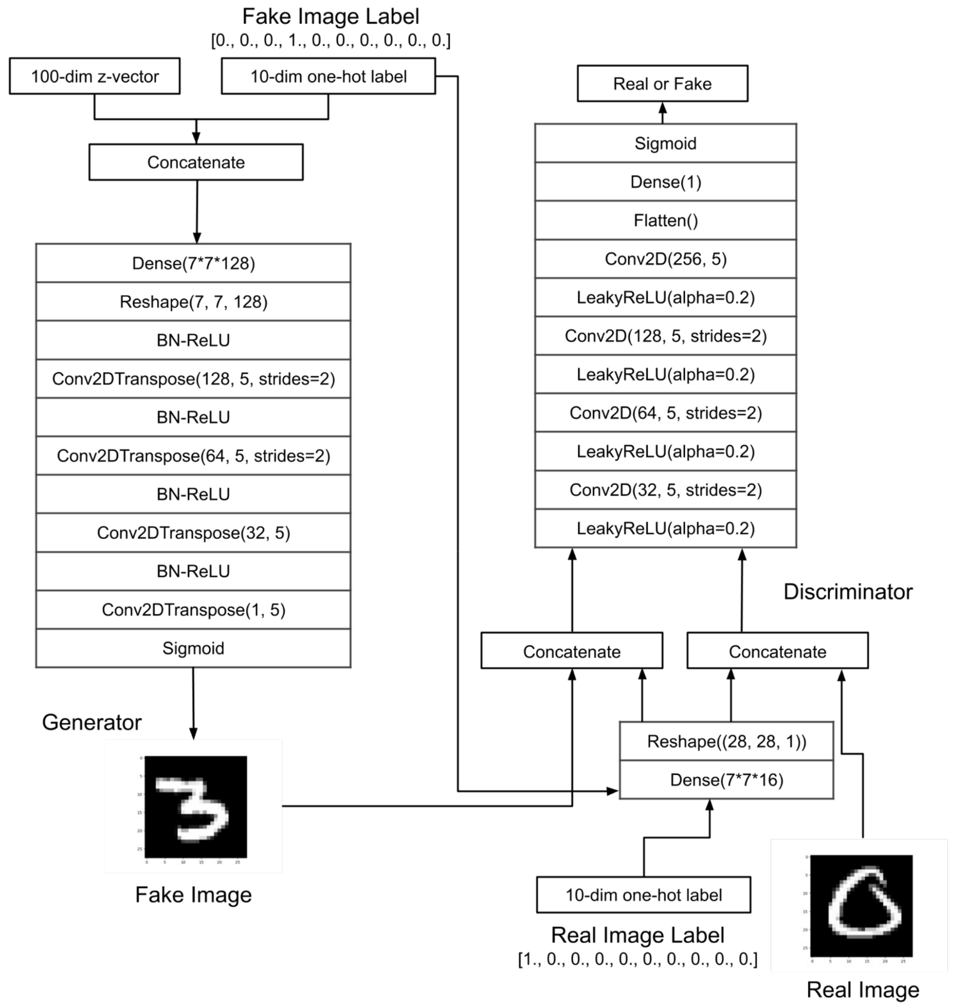

x'
x'
x
y
fake data with label=0.0
real data with label=1.0
one-hot label
Generator generates fake data, ***x'***, conditioned on ***y'***, for Discriminator training
G
input
D
Discriminator is trained against fake & real data conditioned on their respective one-hot labels. Gradients not passed down to Generator.
z
y'
noise
one-hot label
output: probability that input is real
Binary cross-entropy loss

CGAN Generator Training

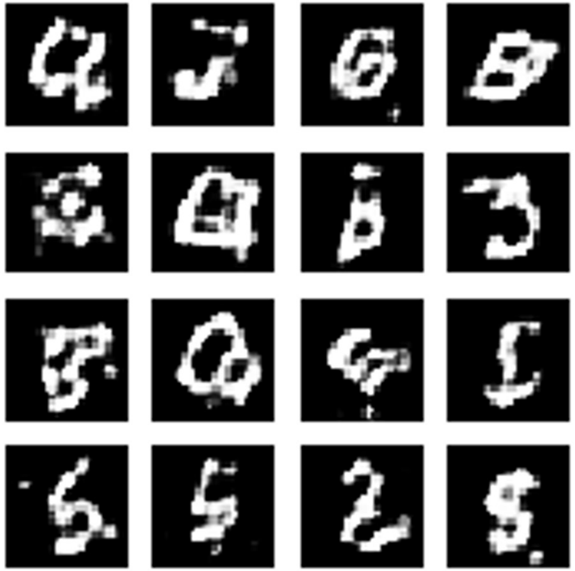

x'
G
D
z
y'
noise
one-hot label
fake data conditioned on y' pretending to be real with label=1.0
input
output: probability that input is real
Binary cross-entropy loss
Generator makes fake data, x', conditioned on y', for Generator training

Discriminator & Generator are trained against conditioned fake data. Discriminator parameters are frozen but gradients are passed down to the Generator.

CGAN in Keras



CGAN outputs conditioned to generate digits 0 to 9



References

Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).

Advanced Deep Learning with TF2 & Keras by Rowel Atienza : https://amzn.to/2wotTnN

End