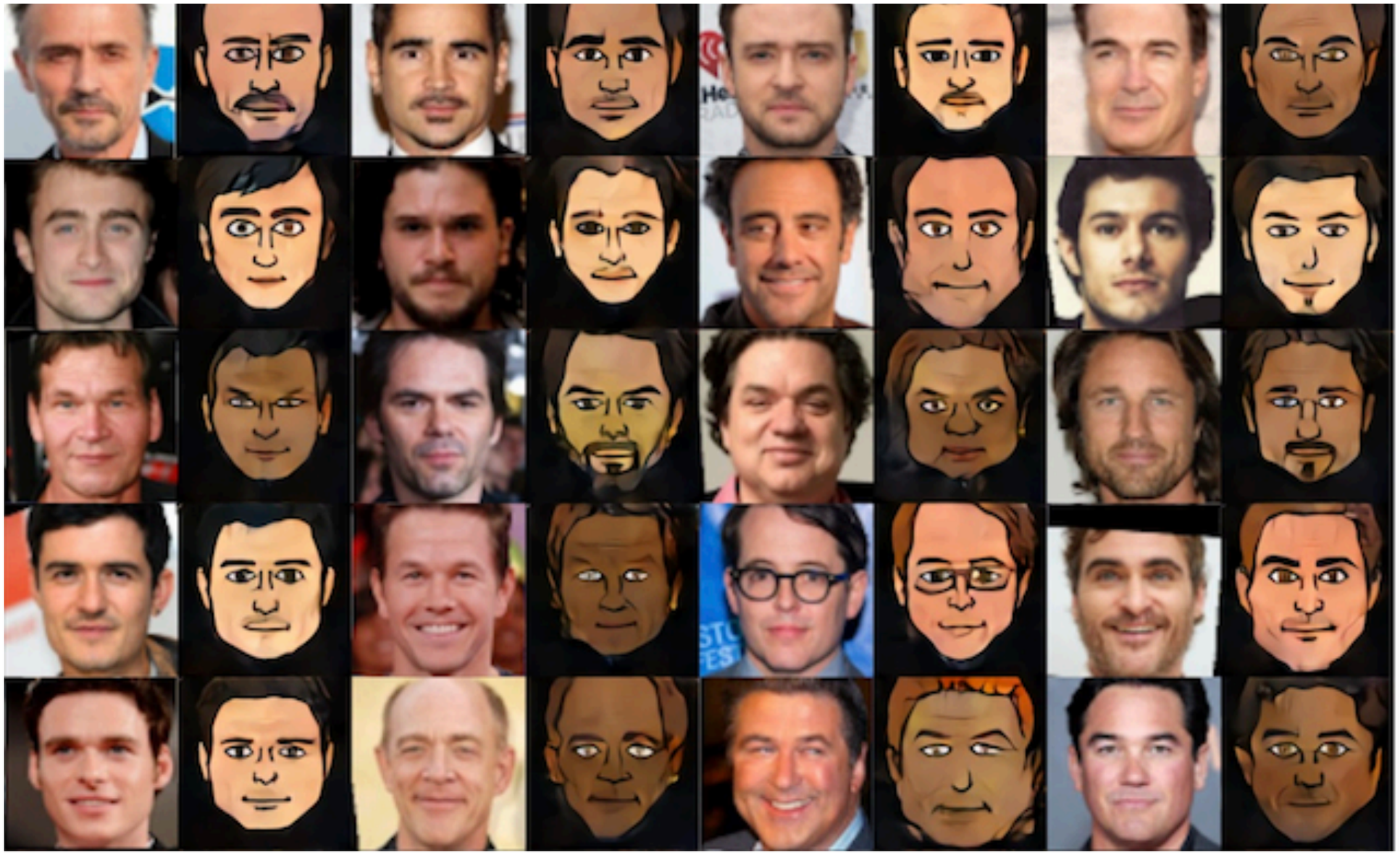


Generative Adversarial Networks

Computer Vision

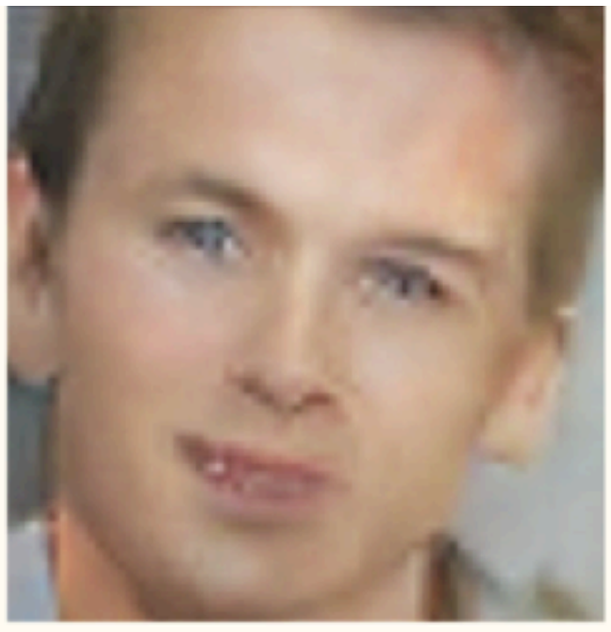
Carnegie Mellon University (Kris Kitani)

Progress of Face Image Generation

[Brundage et al 2018]

2014

[Goodfellow et al.]



2015

[Radford et al.]

2016

[Liu & Tunzel]

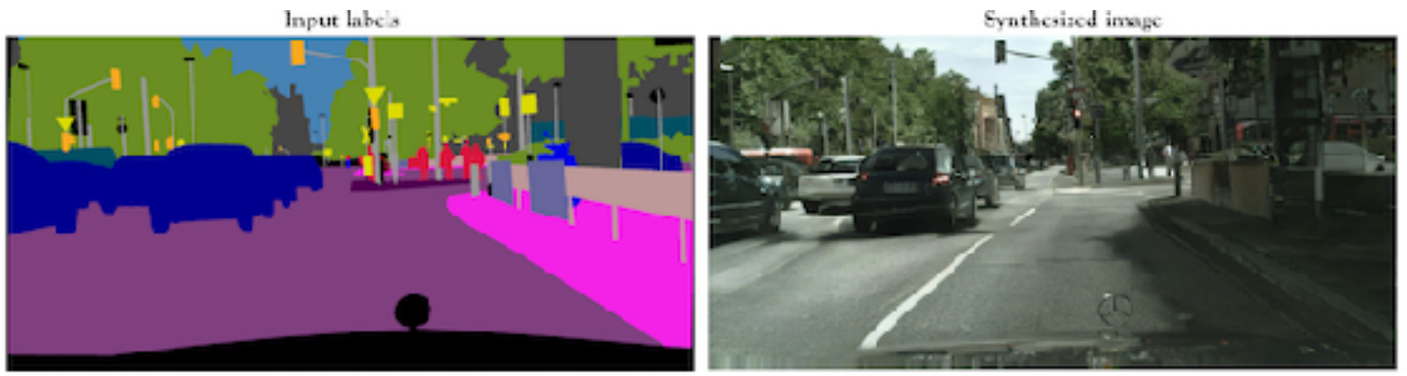
2017

[Karras et al.]

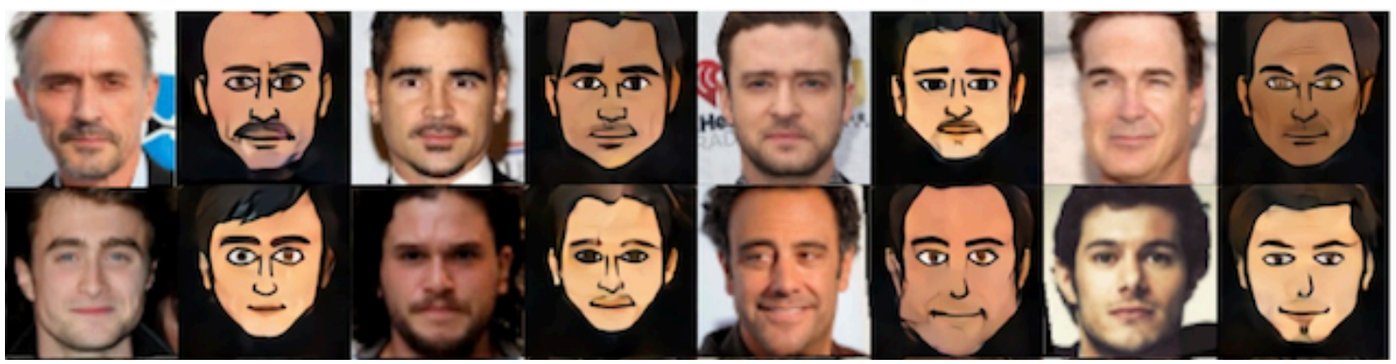
BigGAN [Brock et al. 18]

[Jin et al. 17]

CycleGAN [Zhu et al. 17]



[Wang et al. 17]



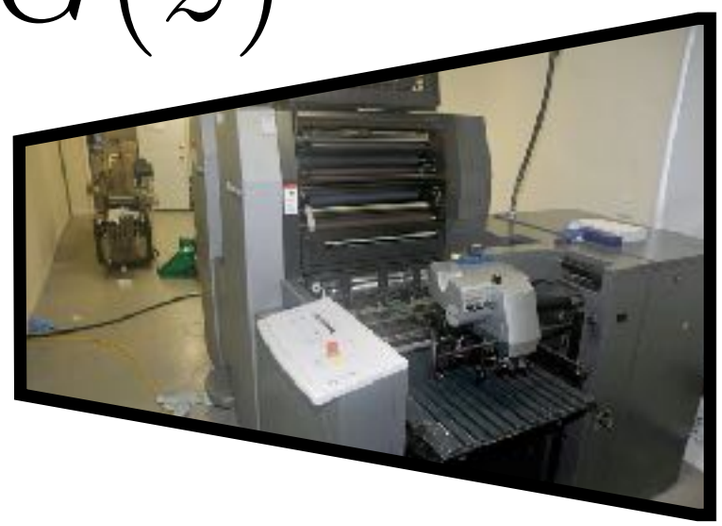
[Taigaman et al. 16]

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Counterfeiters trying to generate fake money

Police trying to detect fake currency

$G(z)$



‘Generator’

x true

x' fake

$D(x)$

‘Discriminator’

over time…

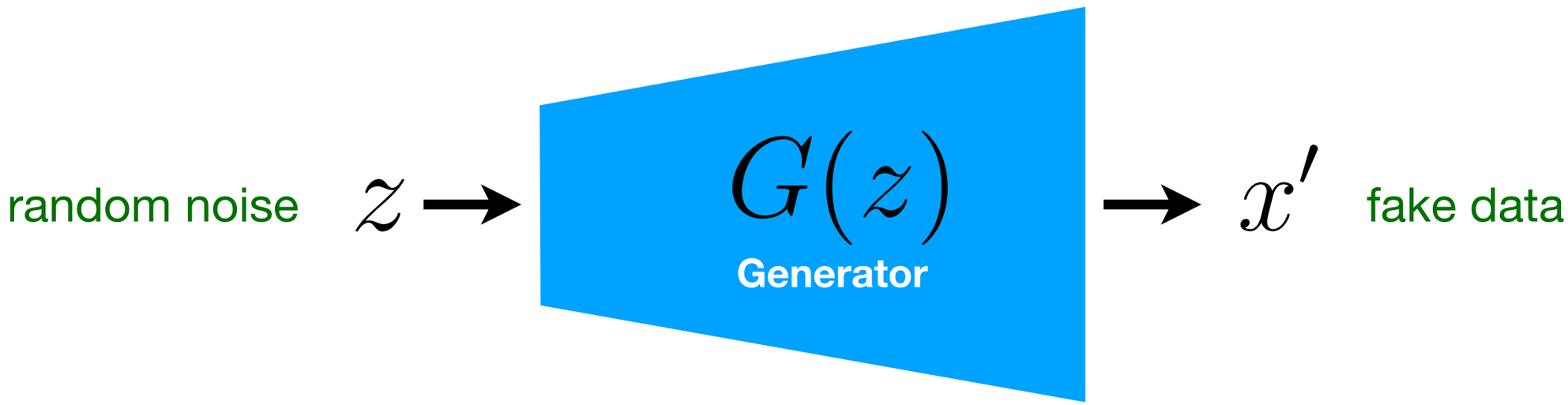
Counterfeiters get better to avoid getting caught

Police get better at identifying fakes

Motivating the Discriminator

How would you score a Generator?

Assume someone gives you a generator:



Generator takes noise and transforms it to a data point (e.g., image, face, trajectory)

$$z \sim q(z) \qquad\qquad x' \sim G(z)$$

If the generator is very good, the fake data should look really ‘similar’ to the true data

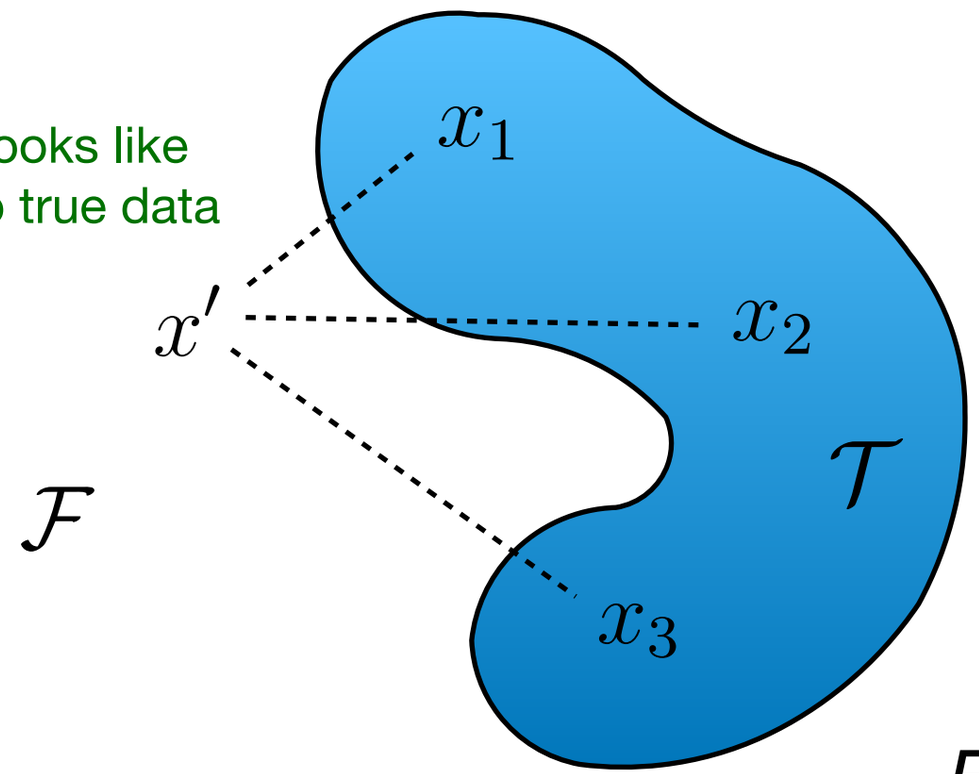
How would you score a Generator?

If the generator is very good, the fake data should look really ‘similar’ to the true data

How should we score a generator?

We could **define** some **distance function**…

Heuristic: Fake data looks like true data if it is close to true data



But what does is mean for a data point to look similar to true data?

But we could also learn it …

Learning a Distance Function

Given:

$$\mathcal{T} = \{x_1, \ldots, x_N\} \qquad \mathcal{F} = \{x'_1, \ldots, x'_N\}$$

True data

Fake data

We want to learn: $D(x; \theta_D) \in [0, 1]$

(inverse) distance function

fake real

How would you learn such a distance function?

(supervised learning)

Learning a Distance Function

Given:

$$\mathcal{T} = \{x_1, \ldots, x_N\} \qquad \mathcal{F} = \{x'_1, \ldots, x'_N\}$$

True data

Fake data

We want to learn:

$$D_\theta(x) = \begin{cases} 1 & \text{x is true} \\ 0 & \text{otherwise} \end{cases}$$

Solve this:

$$\max_\theta \left\{ \sum_{x \in \mathcal{T}} [D_\theta(x)] + \sum_{x' \in \mathcal{F}} [1 - D_\theta(x')] \right\}$$

Should be 1 for true data

Should be 1 for fake data…

since this term should be zero

For all true data

For all fake data

If D is differentiable, it can be learned with gradient descent.

Learning a Discriminator Function

Given:

$$\mathcal{T} = \{x_1, \ldots, x_N\} \qquad \mathcal{F} = \{x'_1, \ldots, x'_N\}$$

True data

Fake data

We want to learn: $$D_\theta(x) = \begin{cases} 1 & \text{x is true} \\ 0 & \text{otherwise} \end{cases}$$

Solve this:

$$\max_\theta \left\{ \sum_{x \in \mathcal{T}} [D_\theta(x)] + \sum_{x' \in \mathcal{F}} [1 - D_\theta(x')] \right\}$$

Should be 1 for true data

Should be 1 for fake data…

since this term should be zero

For all true data

For all fake data

Let's call this the '**Discriminator**'

Because it differentiates between true and false data

If we had a generator and some true data,
now we know how to train a discriminator

$$\max_{\theta} \left\{ \sum_{x \in \mathcal{T}} [D_\theta(x)] + \sum_{x' \in \mathcal{F}} [1 - D_\theta(x')] \right\}$$

$$\max_{\theta} \left\{ \sum_{x \sim p(x)} [D_\theta(x)] + \sum_{z \sim q(z)} [1 - D_\theta(G(z))] \right\}$$

alternative notation (explicit generator)

Motivating the Generator

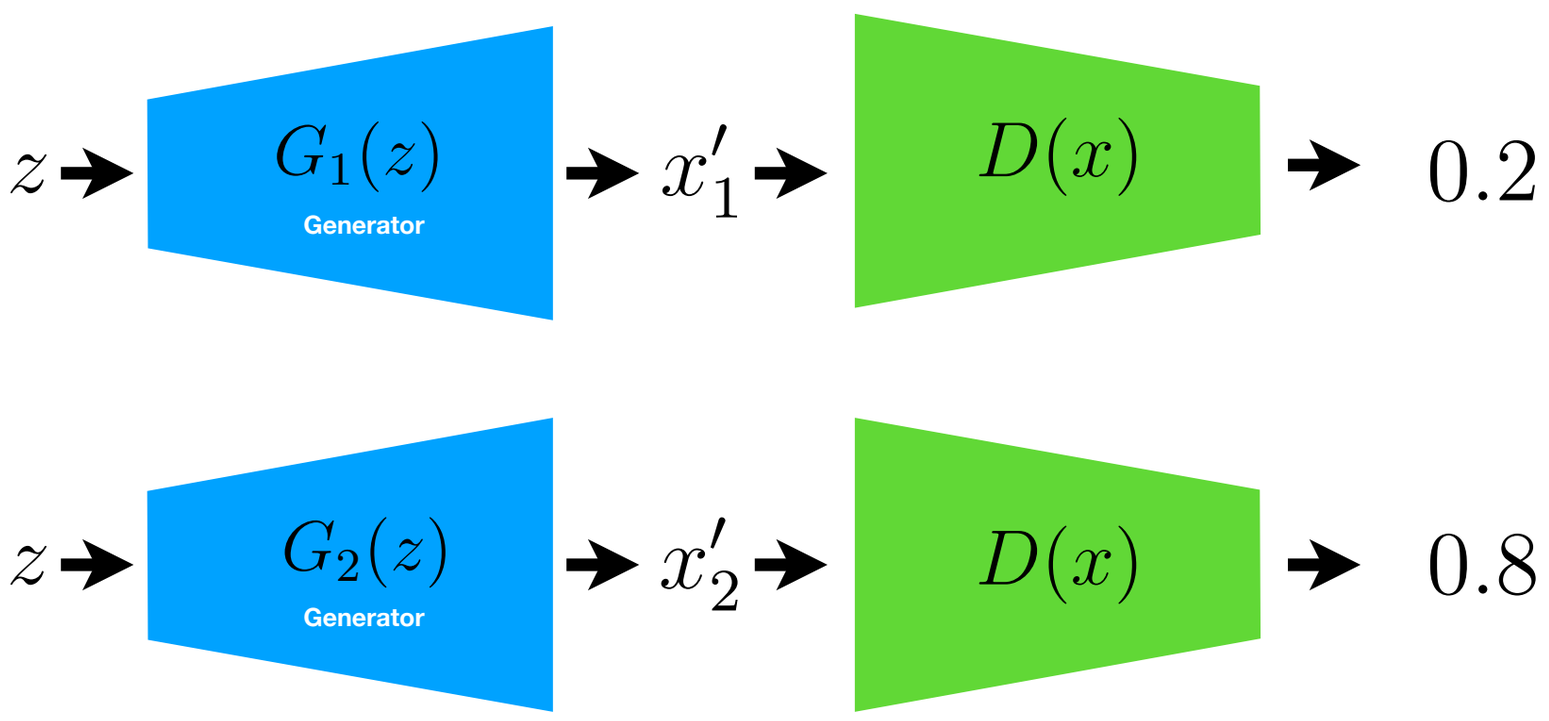
How would you score a Generator?

Assume someone gives you a **discriminator**:

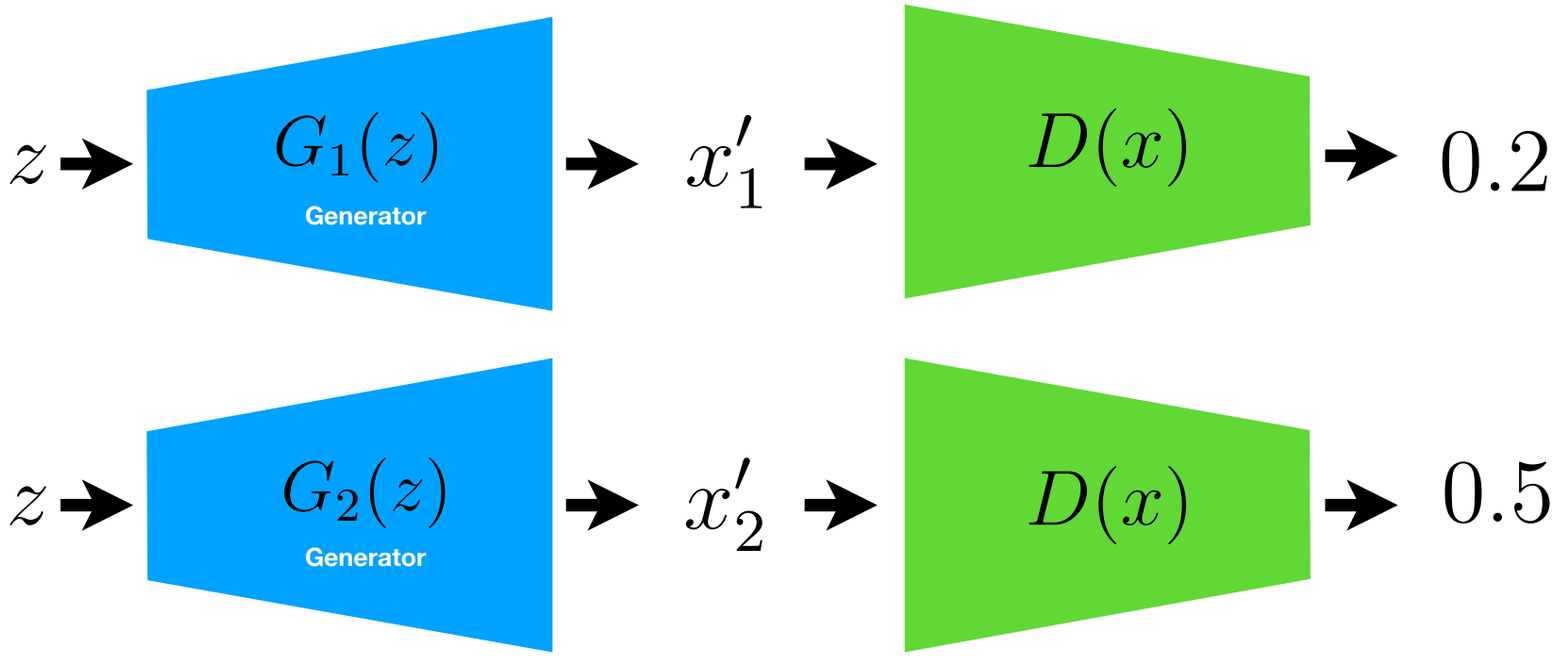
$$D_\theta(x) = \begin{cases} 1 & \text{x is true} \\ 0 & \text{otherwise} \end{cases}$$

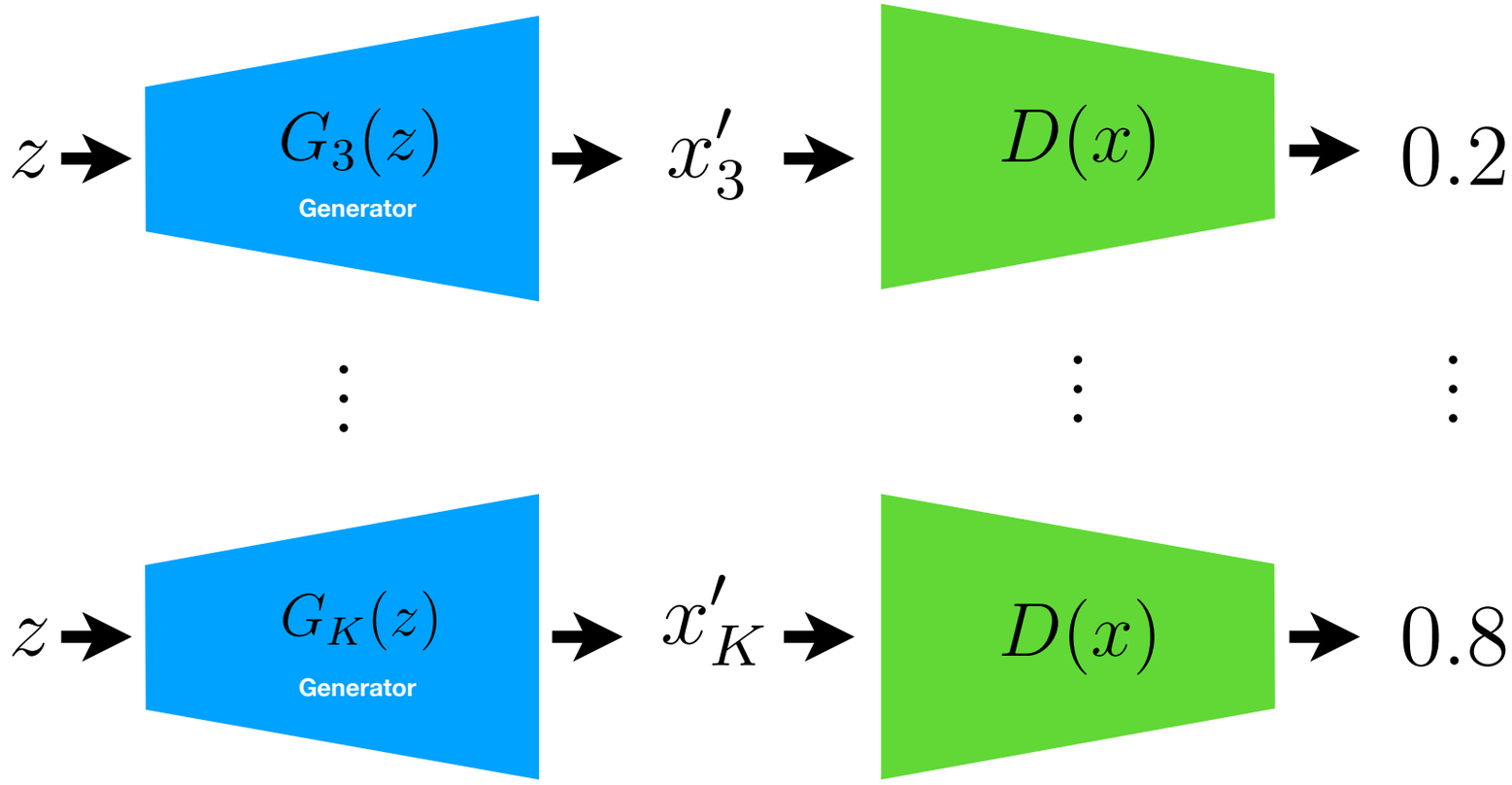
If you had a really good generator,
it should be able to trick the discriminator

Which generator is better? Why?



How would you find the best generator?





Brute Force?
You could get a whole bunch of random generators and keep the best one

But that could take a really long time!

We could also learn it …

Learning a Generator

Given: $D_\theta(x) = \begin{cases} 1 & \text{x is true} \\ 0 & \text{otherwise} \end{cases}$

Could we learn a generator if we had access to only the discriminator function?

(yes, supervised learning)

Learning a Generator

Given: $D_\theta(x) = \begin{cases} 1 & \text{x is true} \\ 0 & \text{otherwise} \end{cases}$

Could we learn a generator if we had access to only the discriminator function?

$$\min_{\phi} \left\{ \sum_{z \sim p(z)} [1 - D(G_\phi(z))] \right\}$$

Should be close to 0 for fake data…

since this term should be close to one

If G is differentiable, it can be learned with gradient descent.

If we had a discriminator,
now we know how to train a generator

$$\min_{\phi} \left\{ \sum_{z \sim p(z)} [1 - D(G_{\phi}(z))] \right\}$$

Generator and true data → Discriminator

$$\max_{\theta} \left\{ \sum_{x \sim p(x)} [D_\theta(x)] + \sum_{z \sim q(z)} [1 - D_\theta(G(z))] \right\}$$

Discriminator → Generator

$$\min_{\phi} \left\{ \sum_{z \sim p(z)} [1 - D(G_\phi(z))] \right\}$$

How do we put these two ideas together?

Formulation

How do we put these two ideas together?

$$\max_{\theta} \left\{ \sum_{x \sim p(x)} [D_\theta(x)] + \sum_{z \sim q(z)} [1 - D_\theta(G(z))] \right\}$$

Learn discriminator, given generator

$$\min_{\phi} \left\{ \sum_{z \sim p(z)} [1 - D(G_\phi(z))] \right\}$$

Learn generator, given discriminator

We learn both the generator and the discriminator at the same time!

$$\min_{\phi} \max_{\theta} \left\{ \sum_{x \sim p(x)} [D_\theta(x)] + \sum_{z \sim q(z)} [1 - D_\theta(G_\phi(z))] \right\}$$

Should be 1 — Fake data

true data — random noise

Should be 0 (discriminator)
Should be 1 (generator)

Formulation in original paper…

$$\min_{\phi} \max_{\theta} \sum_{x \sim p(x)} [\log D_\theta(x)] + \sum_{z \sim q(z)} [\log\{1 - D_\theta(G_\phi(z))\}]$$

1: **function** GAN($\mathcal{T}$)

2: **for** $t = 1, \ldots, T$ **do**

3: Noise $\{z_m\}_{m=1}^{M}$ sampled from $q(z)$

4: True data $\{x_m\}_{m=1}^{M}$ sampled from $\mathcal{T}$

5: $\phi = \phi + \nabla_\phi \frac{1}{M} \sum_m [\log D_\theta(x_m) + \log(1 - D_\theta(G_\phi(z_m)))]$

6: Noise $\{z_n\}_{n=1}^{N}$ sampled from $q(z)$

7: $\theta = \theta - \nabla_\theta \frac{1}{N} \sum_n \log(1 - D_\theta(G_\phi(z_n)))$

Only 6 lines of code!

(if you ignore the millions of gradient updates in line 5 and 7)