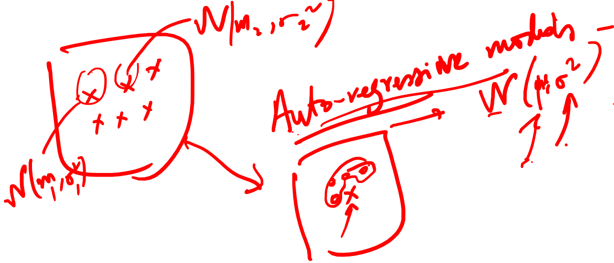
Generative Adversarial Networks

Generative models

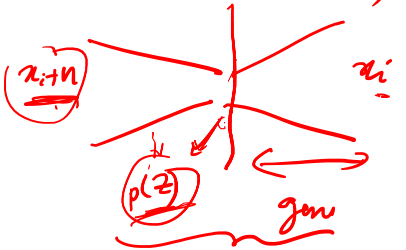


1.) Parametric model
2.) Non-parametric model
histogram, kernel
3.) VAE
4.) GAN → $\cancel{p(x)}$

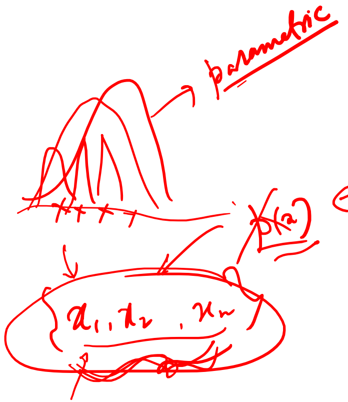
$p(x)$

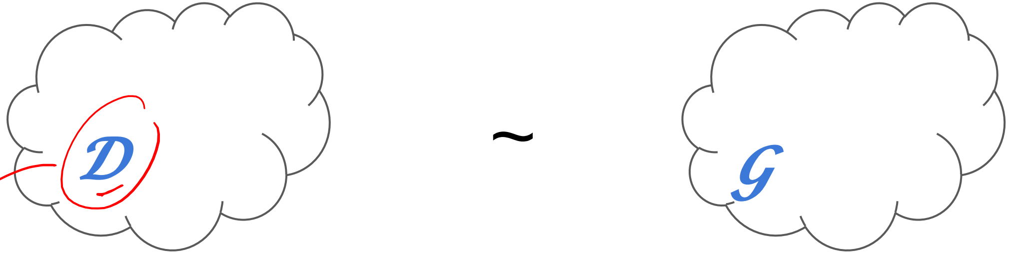
Generate new samples from the distribution of training data

Do not learn just the same samples as that of training data



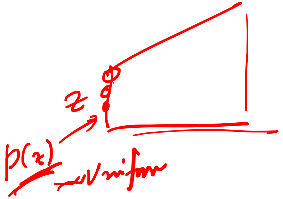
Variational Auto encoder
VAE





desired data distribution

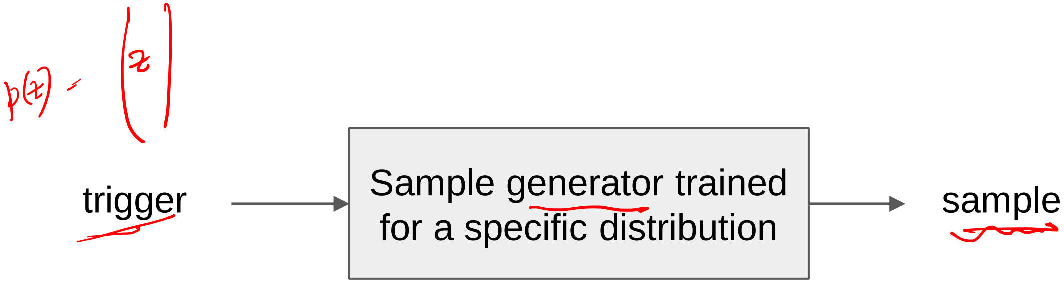
generated distribution



Generative Adversarial Network (GAN)

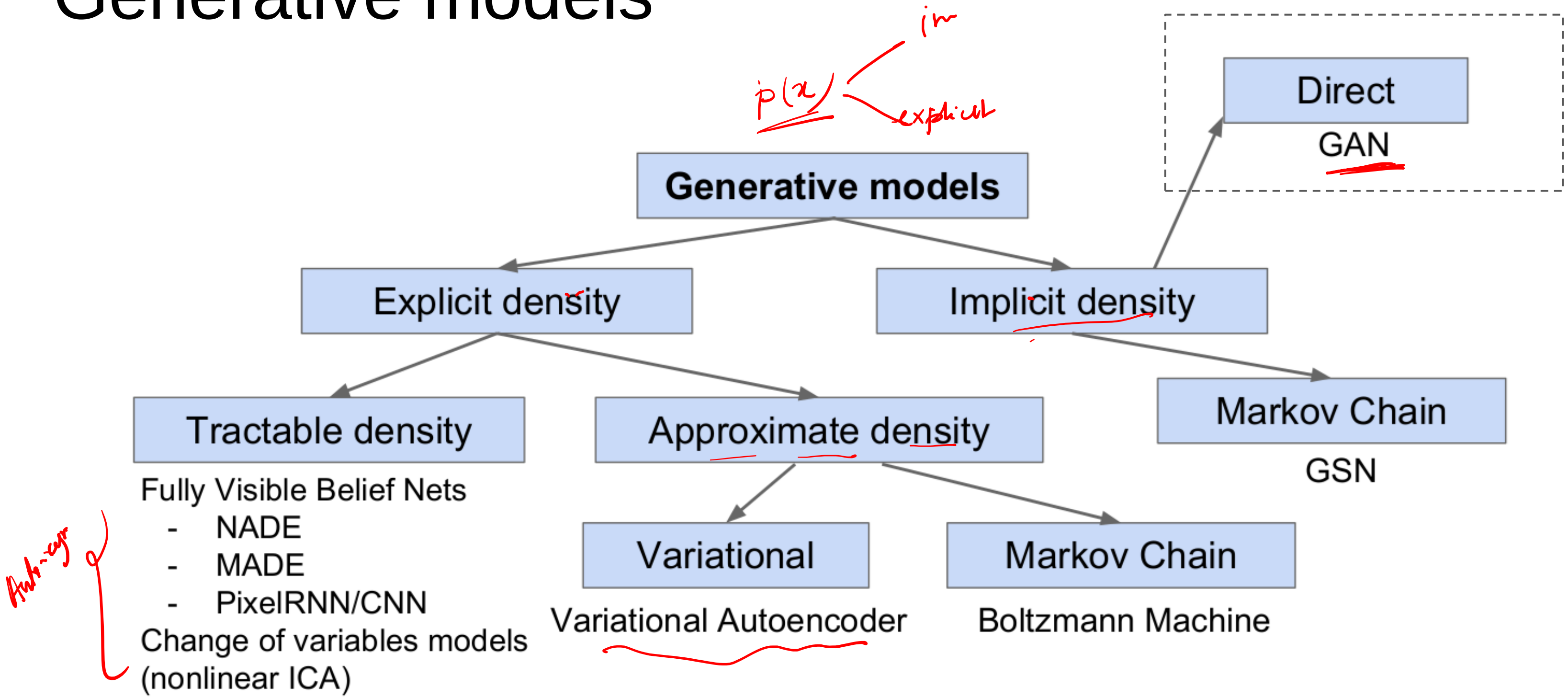
Real distributions are difficult to model as explicit density (by formula)

Instead, generate samples from such complex high-dimensional distributions



Generative Adversarial Network (GAN) does not model explicit density

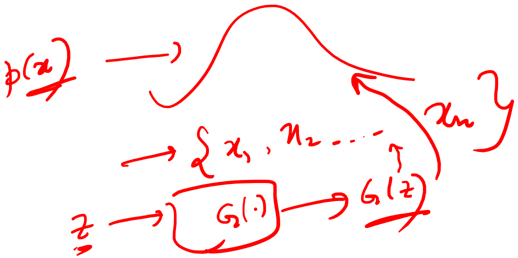
Generative models



GAN

Generative Adversarial Networks

The Generator



The goal is to create a system known as Generator G which will generate “new” samples from a distribution of data

The generator is trained to learn the distribution of data by its own, and is not given the actual distribution (by formula)

To generate different samples, it takes a random vector as its input (trigger)

Generator

uniform $\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix}$

$p(x)$ r.v → uniform

r.v $x = G(z)$

$z_{1\times1}$ uniform → $\boxed{G}$ → $x_{1\times1}$ → $N(0, \sigma^2)$

$f(x)$ → $p(x)$

random vector z $d\times1$ → $\boxed{G}$ → x a sample from the desired distribution

$p(x)$

a number from a uniform distribution in [-A,A]

a vector from a uniform distribution

a number from a Gaussian distribution

a vector from a Gaussian distribution

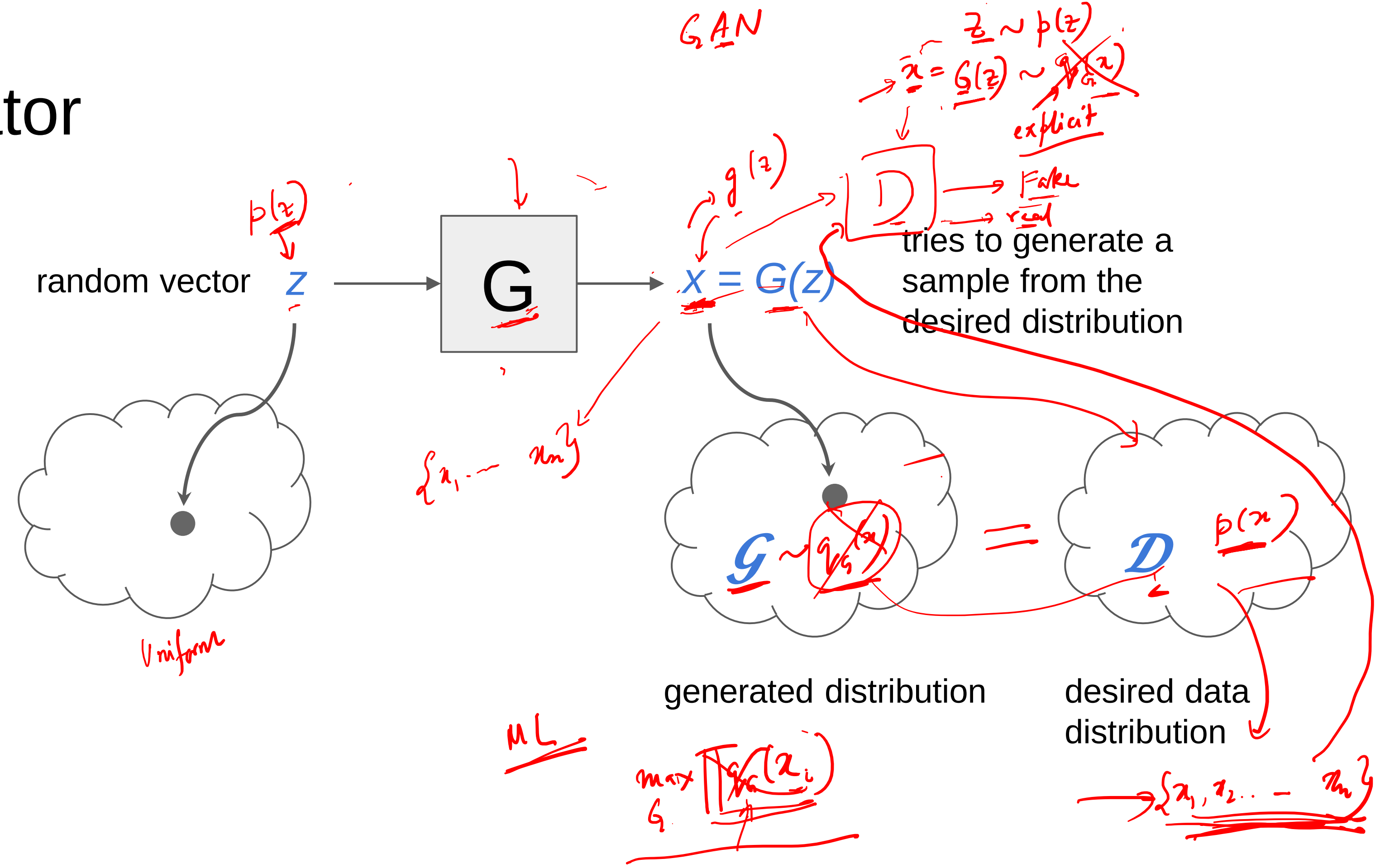
a vector of ones and zeros representing handwritten digits

an image of a building

a natural image → $p(x)$

The generator creates a mapping between these two spaces.

Generator



random vector
z
G
x = G(z)
tries to generate a sample from the desired distribution
generated distribution
desired data distribution
GAN
D
Fake
real
explicit
Uniform
ML
max

Adversarial idea

The generator should not generate the exact same samples used as training data

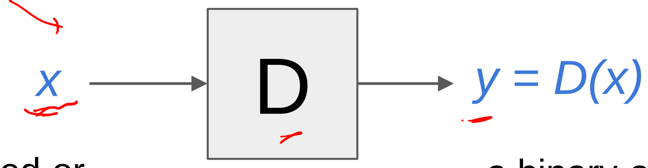
adversarial *adj.- involving or characterized by conflict or opposition*

Instead of training only the generator, we will train an additional system that will tell whether the generator generates samples belonging to the desired distribution, and not whether the generator exactly generates the same training data

Discriminator

G

$\{x_1 \dots x_n\}$



a sample from desired or generative distribution

a binary output that tells whether x is a generated sample or a sample from the data distribution

Real

$D(x) = 1$ if $x \in \{x_1 \dots x_n\}$

$D(x) = 0$ if $x \in G(z)$

$D(G(z)) = 0 \rightarrow$ Fake

$\{x_1 \dots x_n\}$

D outputs 1 if the sample is from the data distribution $\mathcal{D}$

- it says the sample is "*real*"

D outputs 0 if the sample is from the generated distribution $\mathcal{G}$

- it says the sample is "*fake*"

Two player game

$G(z)$
$\|$

$$z \rightarrow \boxed{G} \rightarrow x \rightarrow \boxed{D} \rightarrow y = D(G(z))$$

$D(G(z)) = 0$

G tries to make D(G(z)) near 1
-- "I am not fake"

$D(G(z)) = 1$

D tries to make D(G(z)) near 0
-- "You are fake"

Data

$\{x_1, \dots, x_n\}$

$$x \rightarrow \boxed{D} \rightarrow y = D(x) = 1$$

D tries to output 1
-- "You are real"

Adversarial framework

$0 \le D(x) \le 1$

Data $\{x_1 \dots x_n\}$ $p_{data}(x)$

$\max_D \mathbb{E}_{x \sim p_{data}} [\log(D(x))]$

Discriminator should classify real as 1 $D(x)=1$

$$\max_D \quad \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]$$

D should maximize log D(x) if x comes from data ("real")

Dis. should classify generated data as 0

$$\max_D \quad \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

D should maximize log (1-D(x)) if x comes from generator ("fake")

$D(G(z)) = 0$

$\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$

$$\min_G \quad \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

$\max_G \mathbb{E}_z \log(D(G(z)))$

G should minimize log (1-D(x)) if x comes from generator (G competes by saying "I am real")

$$\min_G \max_D \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{V(D,G)}$$

I. Goodfellow et al. "Generative Adversarial Nets" 2014

Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

$G(z^{(i)})$

  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

real data

  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

Opt the Discriminator

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

Generator

change G ~ Disc opt

$G \sim p(x)$

$D = 0.5$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

I. Goodfellow et al. "Generative Adversarial Nets" 2014

Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

UPDATE DISCRIMINATOR

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by $\boxed{\text{ascending}}$ its stochastic gradient:

maximize

D says generated G(z) is fake, i.e. push D(G(z)) to 0

D says data x is real, i.e. push D(x) to 1

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\boxed{\log D\left(\boldsymbol{x}^{(i)}\right)} + \boxed{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)} \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

I. Goodfellow et al. “Generative Adversarial Nets” 2014

Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

UPDATE GENERATOR

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

minimize

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \boxed{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}.$$

G says generated G(z) is real, i.e. push D(G(z)) to 1

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

I. Goodfellow et al. “Generative Adversarial Nets” 2014

Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by $\boxed{\text{descending}}$ its stochastic gradient:

minimize

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \boxed{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}.$$

Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, log(1 − D(G(z))) saturates.

$D(G(z)) \approx 0$

G says generated G(z) is real, i.e. push log (1-D(G(z))) to -inf

end for

The gradient-based updates can use any standard g[...] tum in our experiments.

Decrease the log probability that the discriminator makes the correct prediction that "gen=fake"

I. Goodfellow et al. "Generative Adversarial Nets" 2014

Training

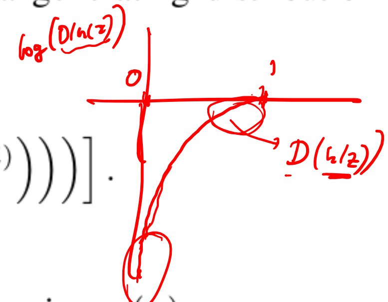
Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$



The generator aims to increase the log probability that the discriminator makes a mistake,
i.e. do max log D(G(z))

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by $\boxed{\text{ascending}}$ its stochastic gradient:

maximize

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \boxed{\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).}$$

rather than aiming to decrease the log probability that the discriminator makes the correct prediction,
i.e. don't use min log (1 - D(G(z)))

G says generated G(z) is real, i.e. push log D(G(z)) to 0

end for

The gradient-based updates can use any standard g tum in our experiments.

Increase the log probability that the discriminator makes the mistake that "gen=real"

I. Goodfellow et al. "Generative Adversarial Nets" 2014

Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

LOOP
UNTIL YOU ARE SATISFIED WITH THE GENERATOR SAMPLES

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

UPDATE DISCRIMINATOR

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient:

UPDATE GENERATOR

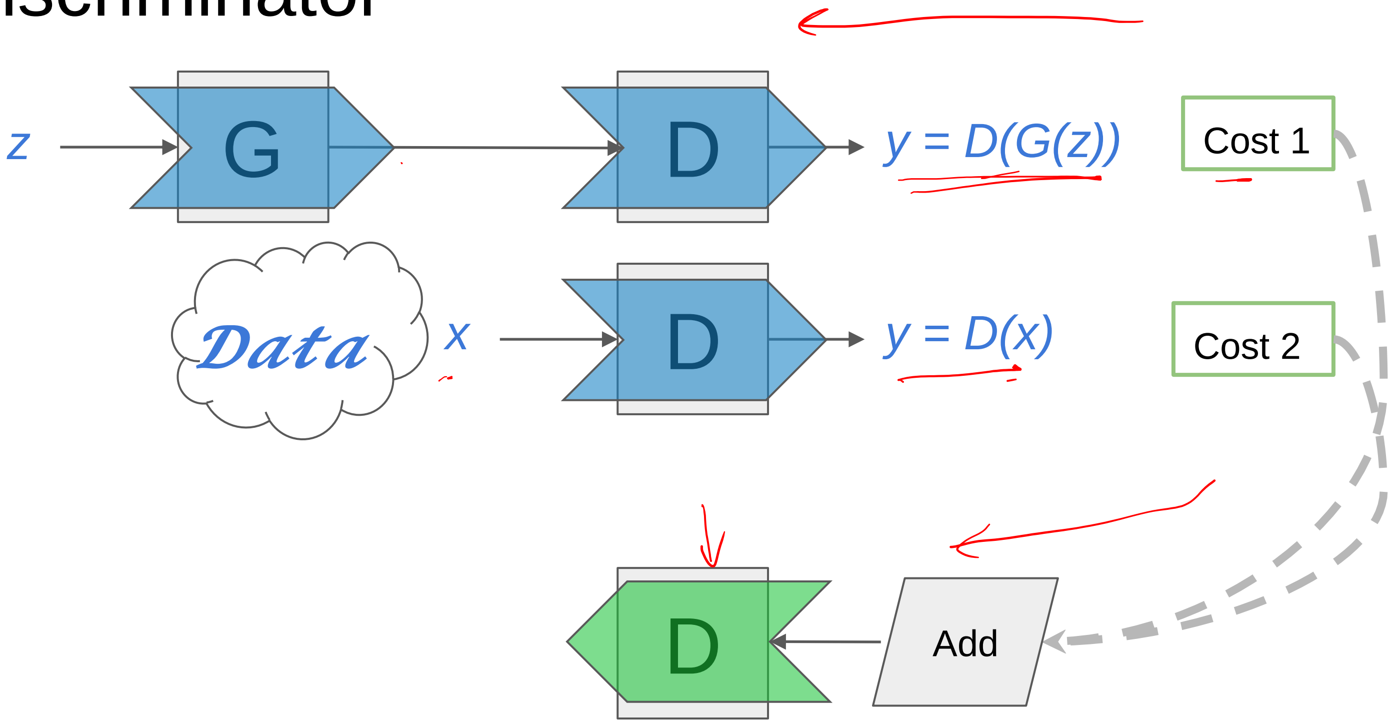
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

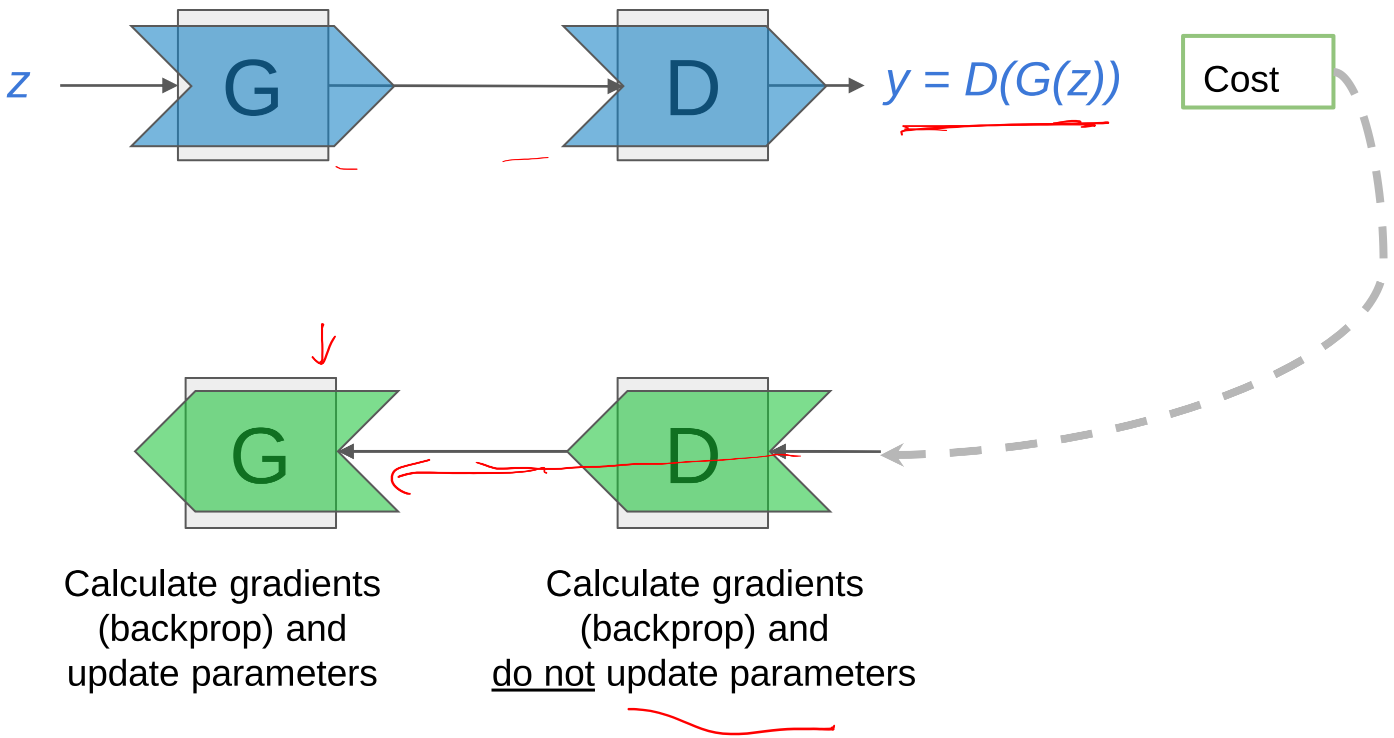
I. Goodfellow et al. “Generative Adversarial Nets” 2014

Update discriminator



Calculate gradients (backprop)
and update parameters

Update generator



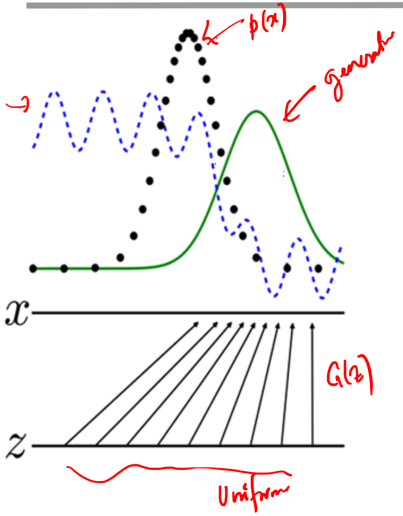
This will be a programming assignment.

Training example illustration

z is sampled from a uniform distribution, and data is Gaussian
Both z and x are single scalars
G and D are multilayer perceptrons

Legend
Data distribution (desired)
Generated distribution
Discriminator output

training both G and D



G maps z to x

I. Goodfellow et al. "Generative Adversarial Nets" 2014

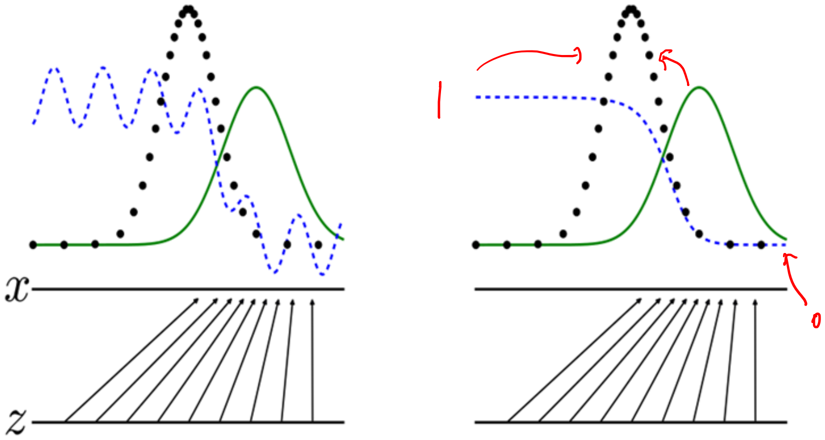
This will be a programming assignment.

Training example illustration

Legend
Data distribution (desired)
Generated distribution
Discriminator output

z is sampled from a uniform distribution, and data is Gaussian
Both z and x are single scalars
G and D are multilayer perceptrons

training both G and D



G maps z to x

G: green (generated dist) slowly moves towards black (data dist)

I. Goodfellow et al. “Generative Adversarial Nets” 2014

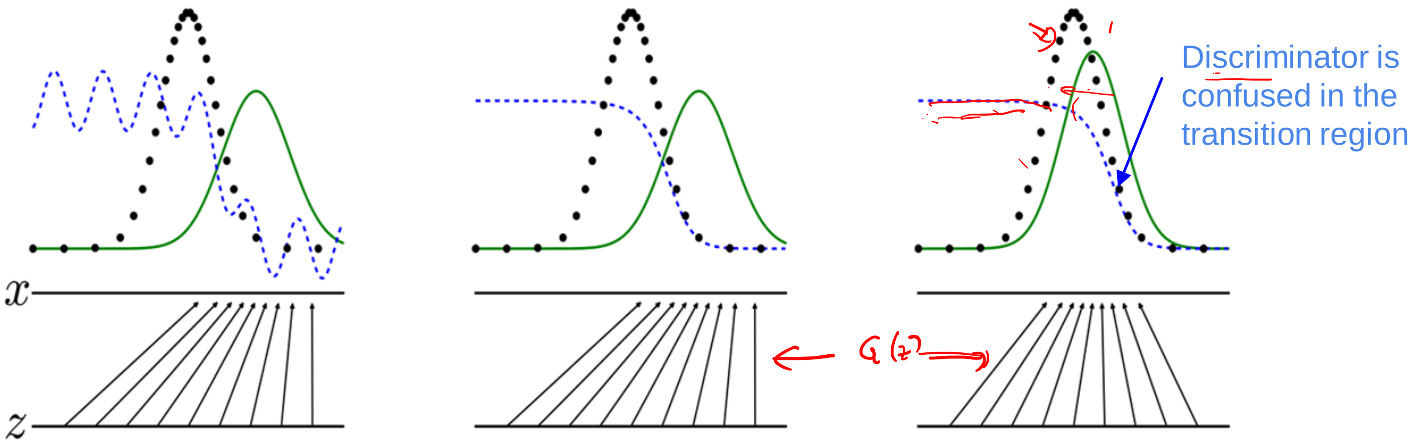
This will be a programming assignment.

Training example illustration

Legend
Data distribution (desired)
Generated distribution
Discriminator output

z is sampled from a uniform distribution, and data is Gaussian
Both z and x are single scalars
G and D are multilayer perceptrons

training both G and D



G maps z to x

G: green (generated dist) slowly moves towards black (data dist)

D: initially, D (blue) classifies green as fake (low), and black as real (high).

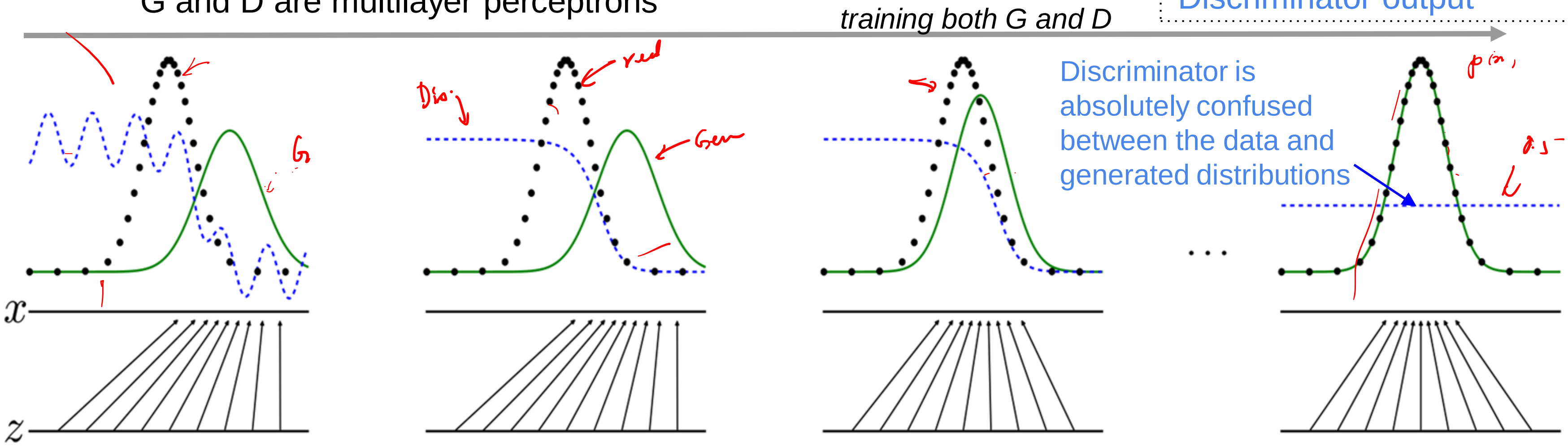
I. Goodfellow et al. "Generative Adversarial Nets" 2014

This will be a programming assignment.

Training example illustration

z is sampled from a uniform distribution, and data is Gaussian
Both z and x are single scalars
G and D are multilayer perceptrons

Legend
Data distribution (desired)
Generated distribution
Discriminator output



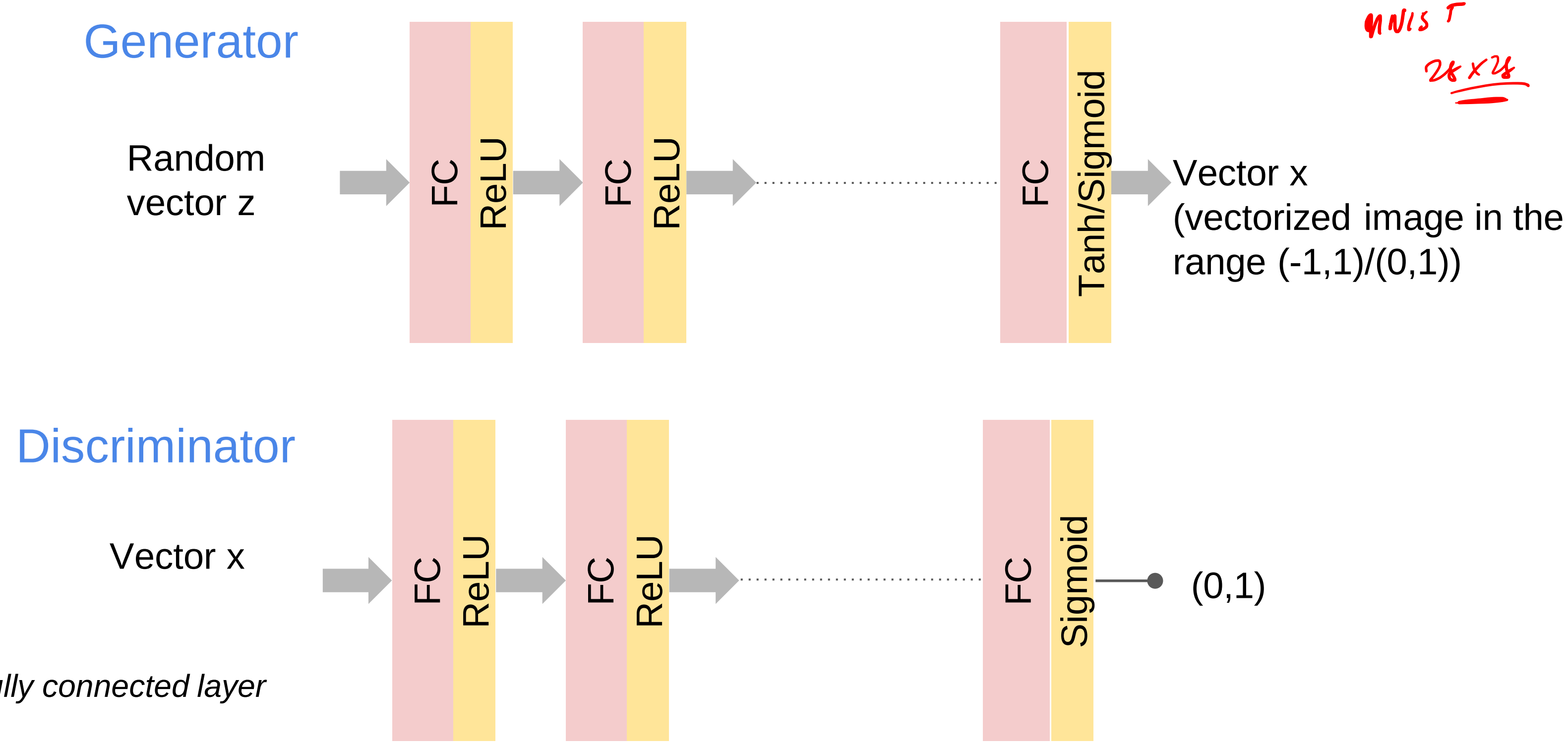
At last, G produces samples from the distribution same as or close to the data distribution

- D finally outputs the same (0.5) for both generated and data distributions.
- *Important:* D does not try to identify the samples from the data distribution. D tries only to differentiate the data distribution and the generated distribution.

After training, the discriminator may be discarded, but some of its features in hidden layers may be used for supervised tasks.

I. Goodfellow et al. "Generative Adversarial Nets" 2014

1D GAN

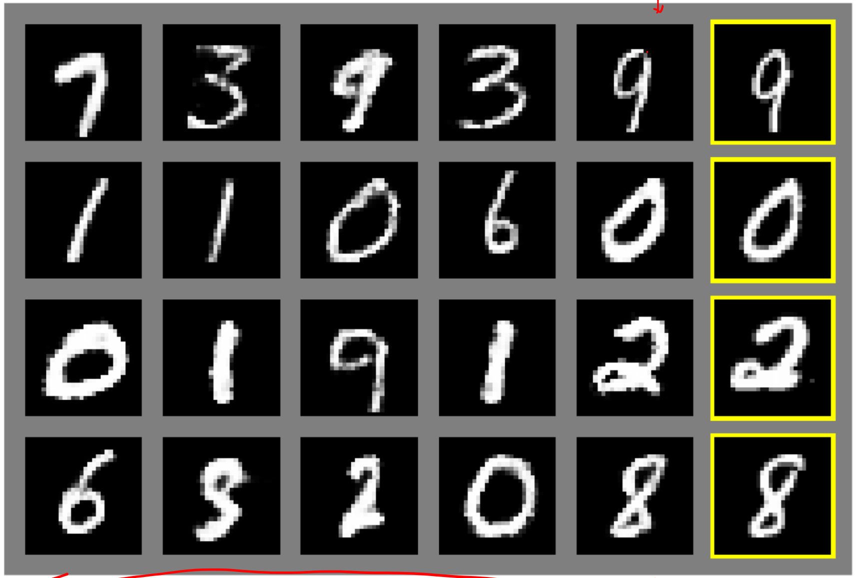


MNIST
28x28

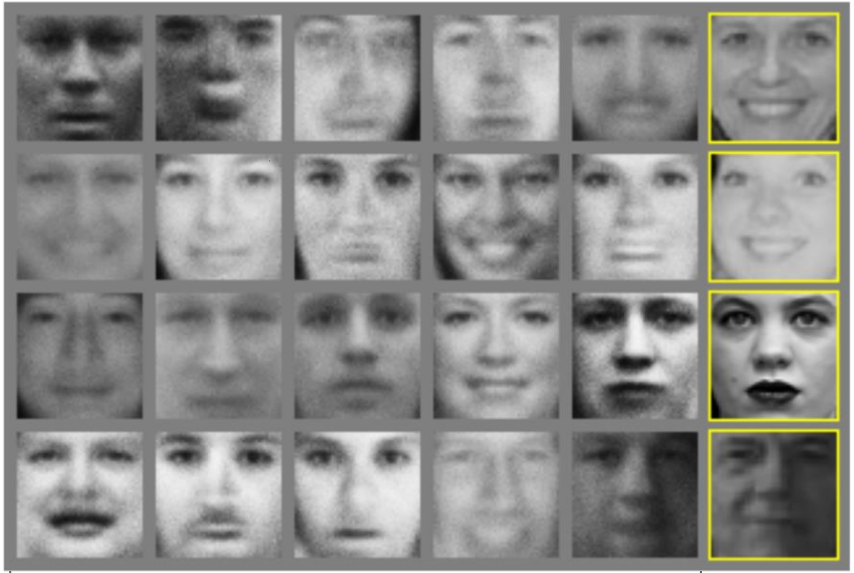
FC - fully connected layer

Sample Results - 1D-GAN to produce 2D images

Trained using MNIST digits dataset



Trained using TFD (Toronto Face Dataset)



Generated samples for different z

Not bad!

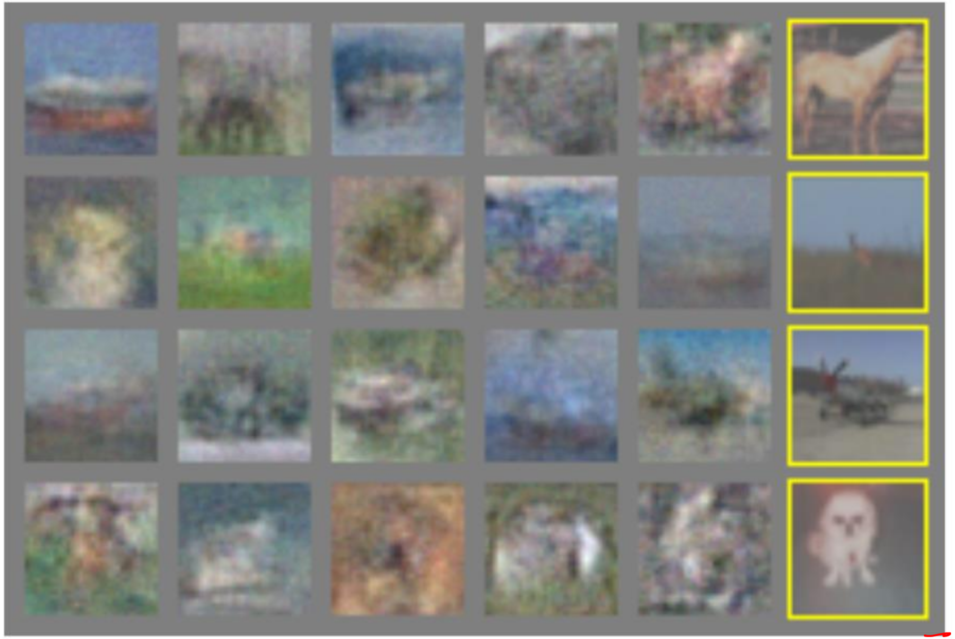
Generated samples for different z

Nearest neighbour from the training data

I. Goodfellow et al. “Generative Adversarial Nets” 2014

Sample Results - 1D-GAN to produce images

CIFAR-10 using fully-connected model for generator



Generated samples for different z

Nearest neighbour
from the training data

Not good!

I. Goodfellow et al. “Generative Adversarial Nets” 2014

DCGAN

Deep convolutional GAN

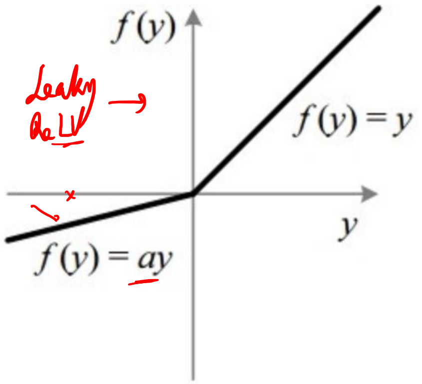
Use convolutional layers with no fully connected layers
To generate better images

Architectural guidelines

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

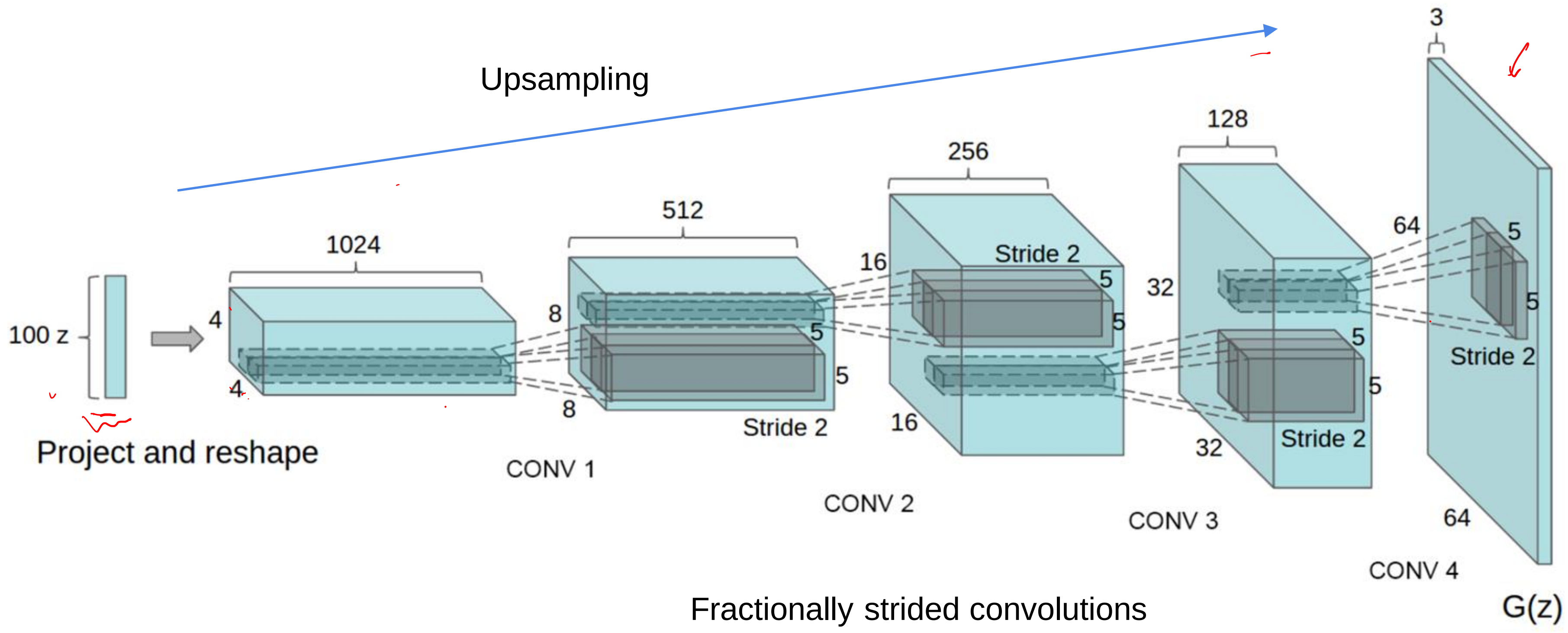
Un pooling

ReLU



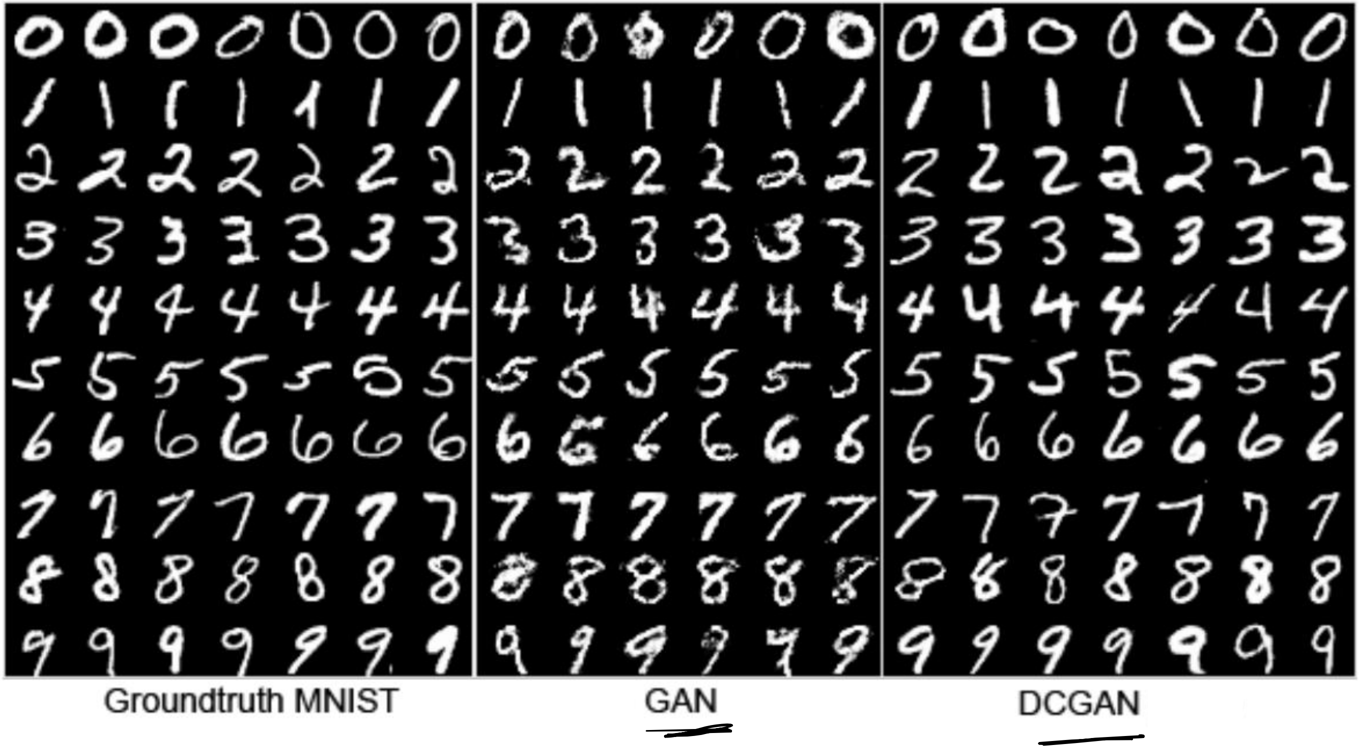
A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Generator in DCGAN



A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Generated outputs using DCGAN - a “child” writes



A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

These faces are
generated.
Not real!

A. Radford et al.
“Unsupervised
Representation Learning with
Deep Convolutional
Generative Adversarial
Networks” 2016

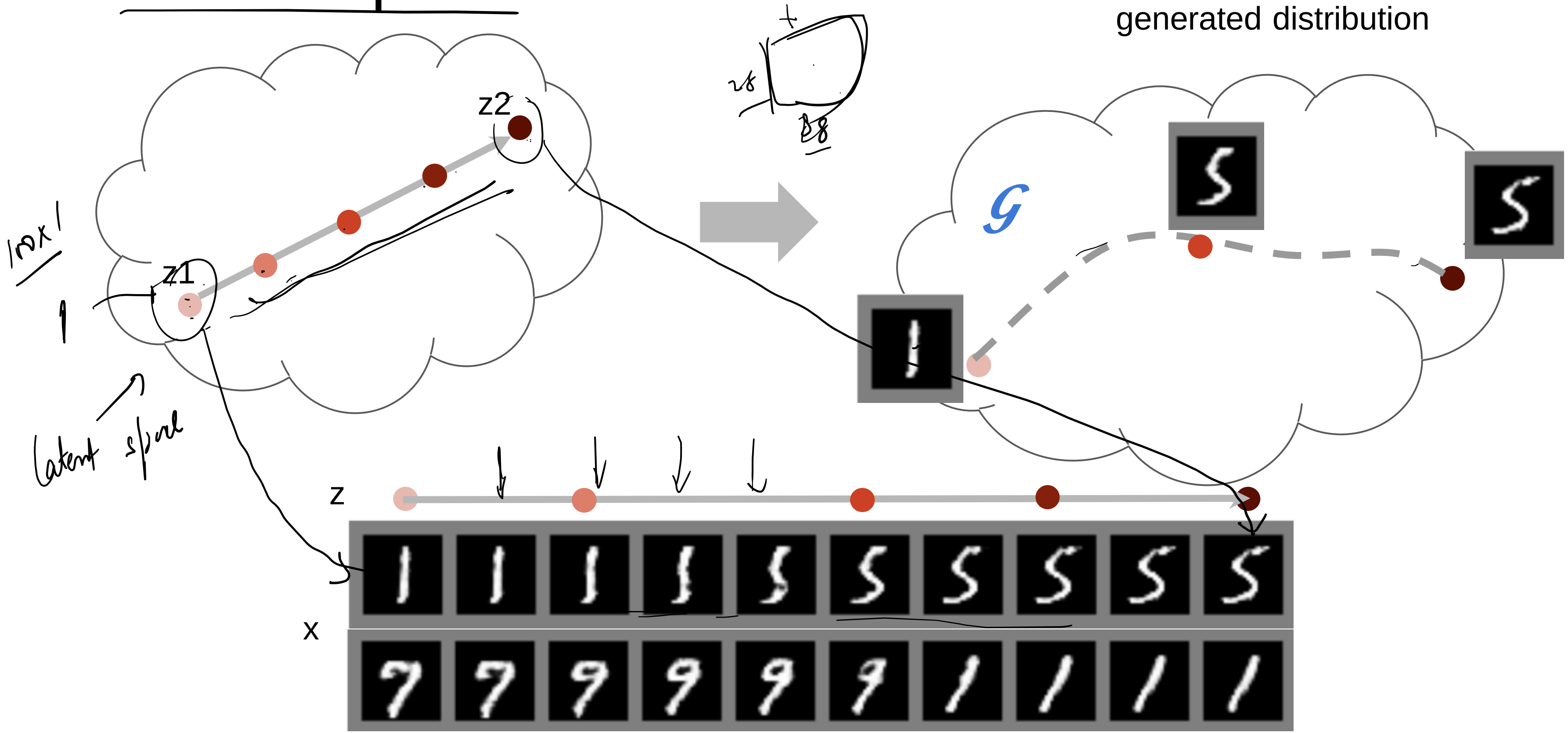
Very recent work - Generate high quality images

Timo Aila, Samuli Laine, Jaakko Lehtinen, "**Progressive Growing of GANs for Improved Quality, Stability, and Variation Tero Karras**" (NVIDIA and Aalto university), submited to ICLR 2018. http://research.nvidia.com/publication/2017-10_Progressive-Growing-of

1024x1024 images trained using CelebA-HQ dataset

These are GAN-generated faces, and are not real.

Vector interpolation for 1D GAN



A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Vector interpolation for DCGAN

A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Vector interpolation for DCGAN

z

x

Face looking left

Face looking right

A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Vector arithmetic in DCGAN

$z^{w\_smile}_1$

$z^{w\_smile}_2$

$z^{w\_smile}_3$

woman smiling

$z^{w\_smile} = \text{mean}(z^{w\_smile}_1, z^{w\_smile}_2, z^{w\_smile}_3)$

A. Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" 2016

Vector arithmetic in DCGAN

$z^{w\_smile}{}_1$

$z^{w\_smile}{}_2$

$z^{w\_smile}{}_3$

$z^w{}_1$

$z^w{}_2$

$z^w{}_3$

$z^m{}_1$

$z^m{}_2$

$z^m{}_3$

$\sum z^{w_{-1}} \xrightarrow{G}$

woman smiling

woman

man

z^{w_smil} $= \text{mean}(z^{w\_smile}{}_1, z^{w\_smile}{}_2, z^{w\_smile}{}_3)$

z^w

z^m

A. Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" 2016

Vector arithmetic in DCGAN

$z^{w\_smile}_1$ $z^{w\_smile}_2$ $z^{w\_smile}_3$

z^w_1 z^w_2 z^w_3

z^m_1 z^m_2 z^m_3

woman smiling	woman	man
z^{w_smil}	z^w	z^m

$= \text{mean}(z^{w\_smile}_1, z^{w\_smile}_2, z^{w\_smile}_3)$

define $z^{m\_smile} = z^{w\_smile} - z^w + z^m$

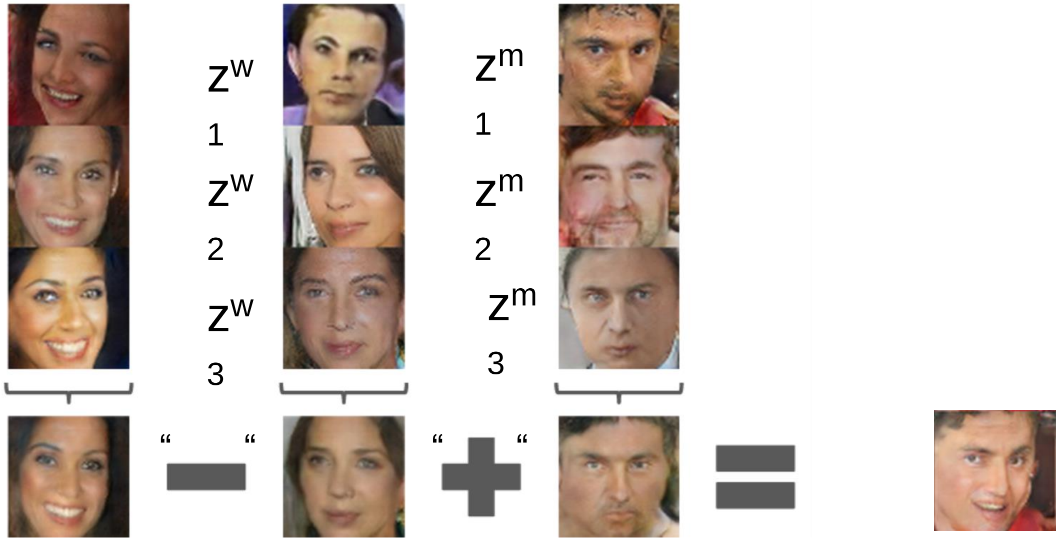
A. Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" 2016

Vector arithmetic in DCGAN

$z^{w\_smile}_1$ z^{w}_1 z^{m}_1

$z^{w\_smile}_2$ z^{w}_2 z^{m}_2

$z^{w\_smile}_3$ z^{w}_3 z^{m}_3



woman smiling | woman | man | man smiling

z^{w_smil} z^{w} z^{m}

$= \text{mean}(z^{w\_smile}_1, z^{w\_smile}_2, z^{w\_smile}_3)$

define $z^{m\_smile} = z^{w\_smile} - z^{w} + z^{m}$

It performs the “operation” of subtracting “woman” from “woman smiling” and adding “man” in the generated distribution

A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Vector arithmetic in DCGAN

$z^{w\_smile}_1$ $z^{w\_smile}_2$ $z^{w\_smile}_3$

z^{w}_1 z^{w}_2 z^{w}_3

z^{m}_1 z^{m}_2 z^{m}_3

z^{m_neut} + small uniform noise to test the neighbours

woman smiling | woman | man | man smiling

z^{w_smil} $= \mathrm{mean}(z^{w\_smile}_1, z^{w\_smile}_2, z^{w\_smile}_3)$

z^{w}

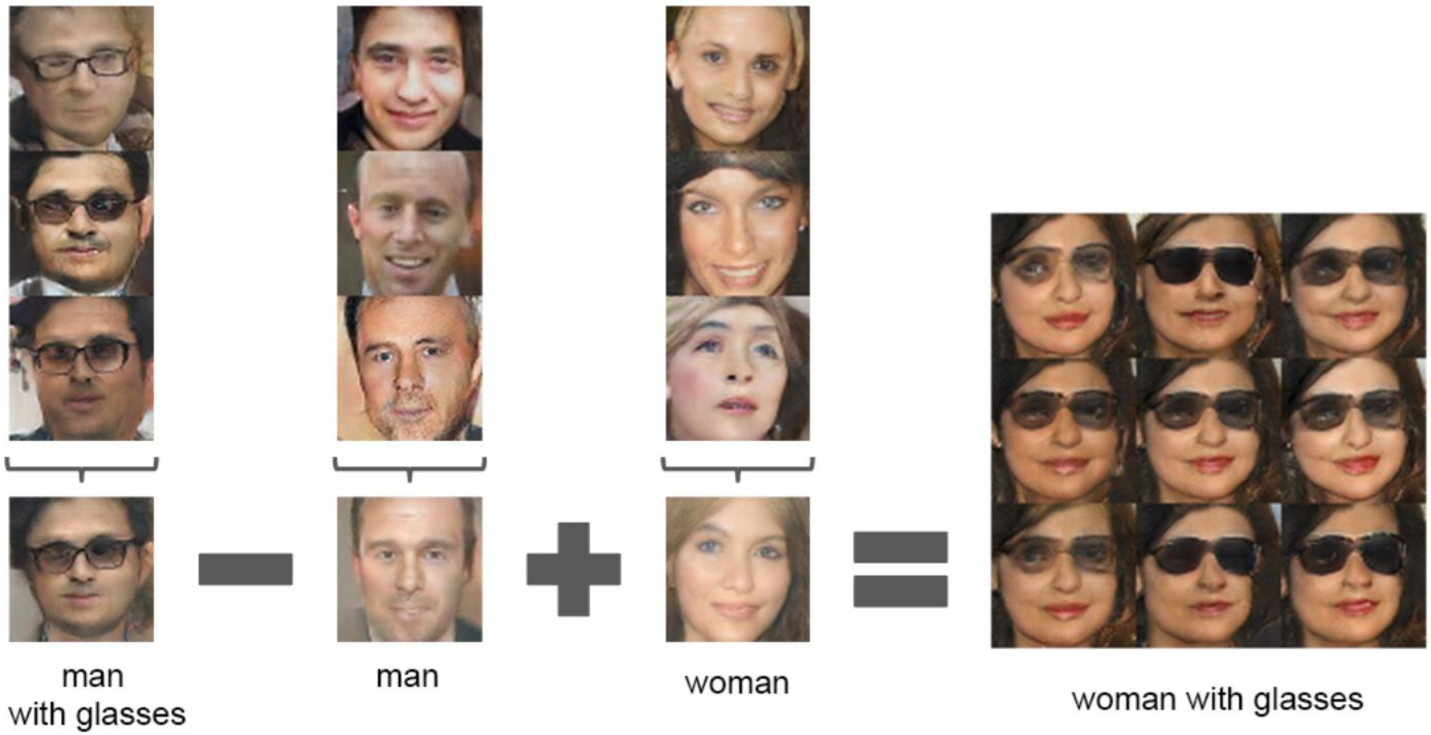
z^{m}

define $z^{m\_smile} = z^{w\_smile} - z^{w} + z^{m}$

It performs the “operation” of subtracting “woman” from “woman smiling” and adding “man” in the generated distribution

A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Vector arithmetic in DCGAN



A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” 2016

Conditional GAN

Conditional GAN

In an unconditioned generative model, there is no control on modes of the data being generated

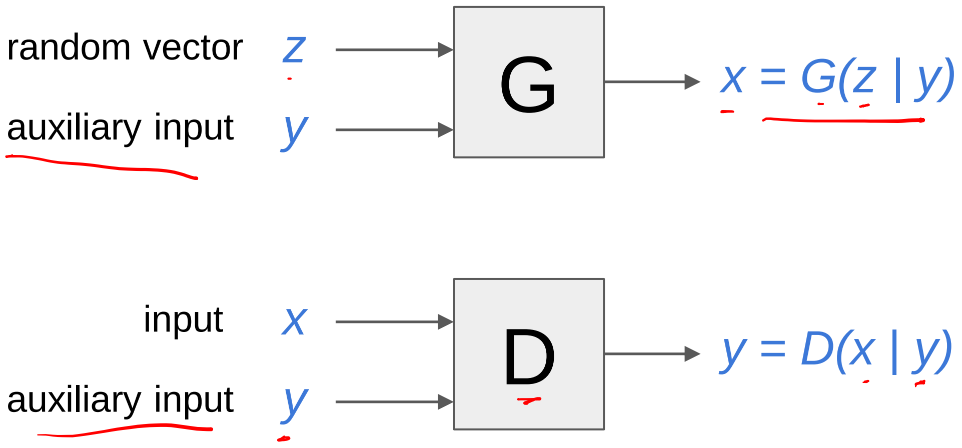
By conditioning the model on additional information, it is possible to direct the data generation process

Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y

y could be any kind of auxiliary information, such as class labels or data from other modalities (e.g. image)

Mirza and Osindero “Conditional Generative Adversarial Nets” 2014

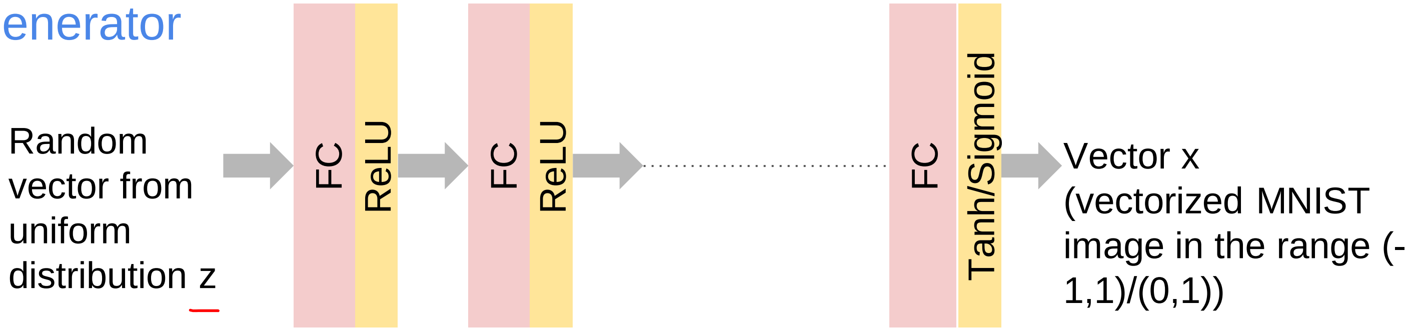
Conditional GAN



$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

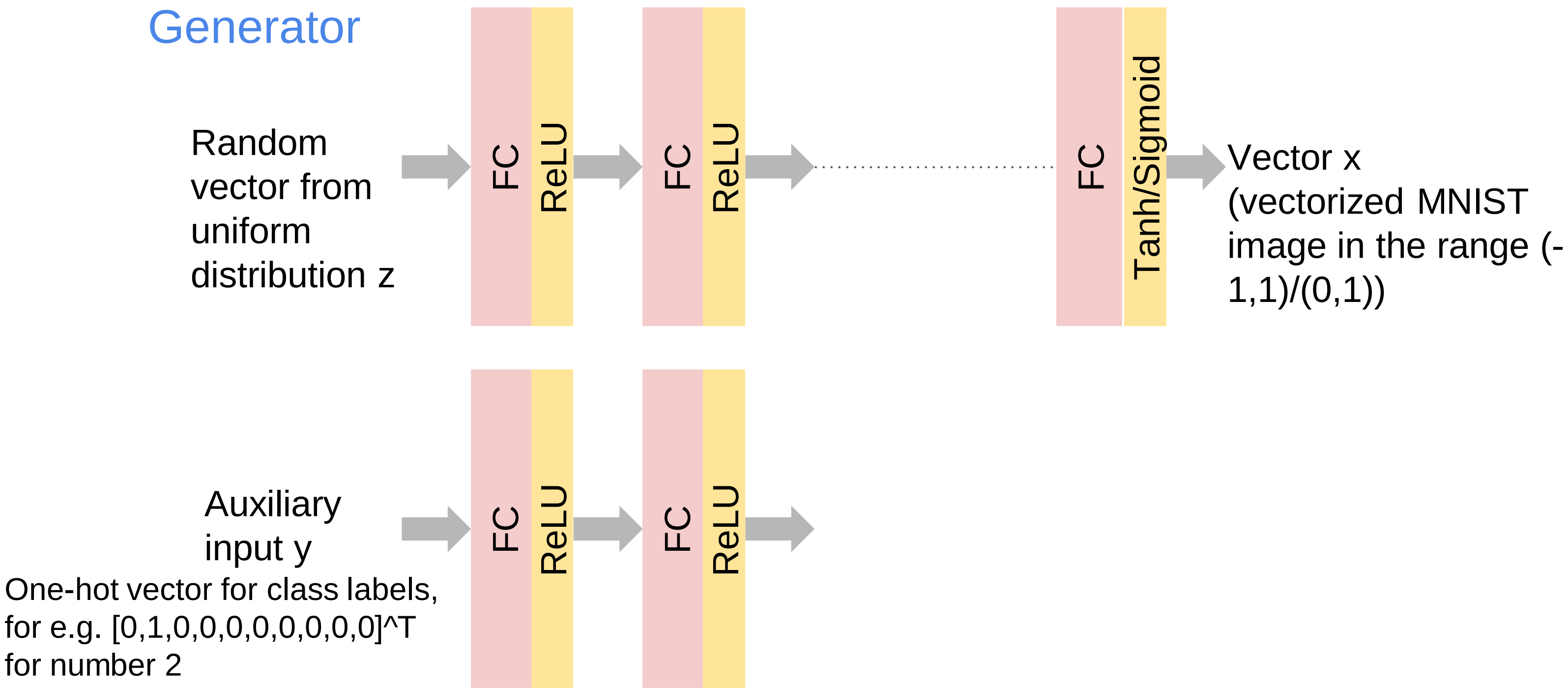
Conditional GAN - MNIST

Generator



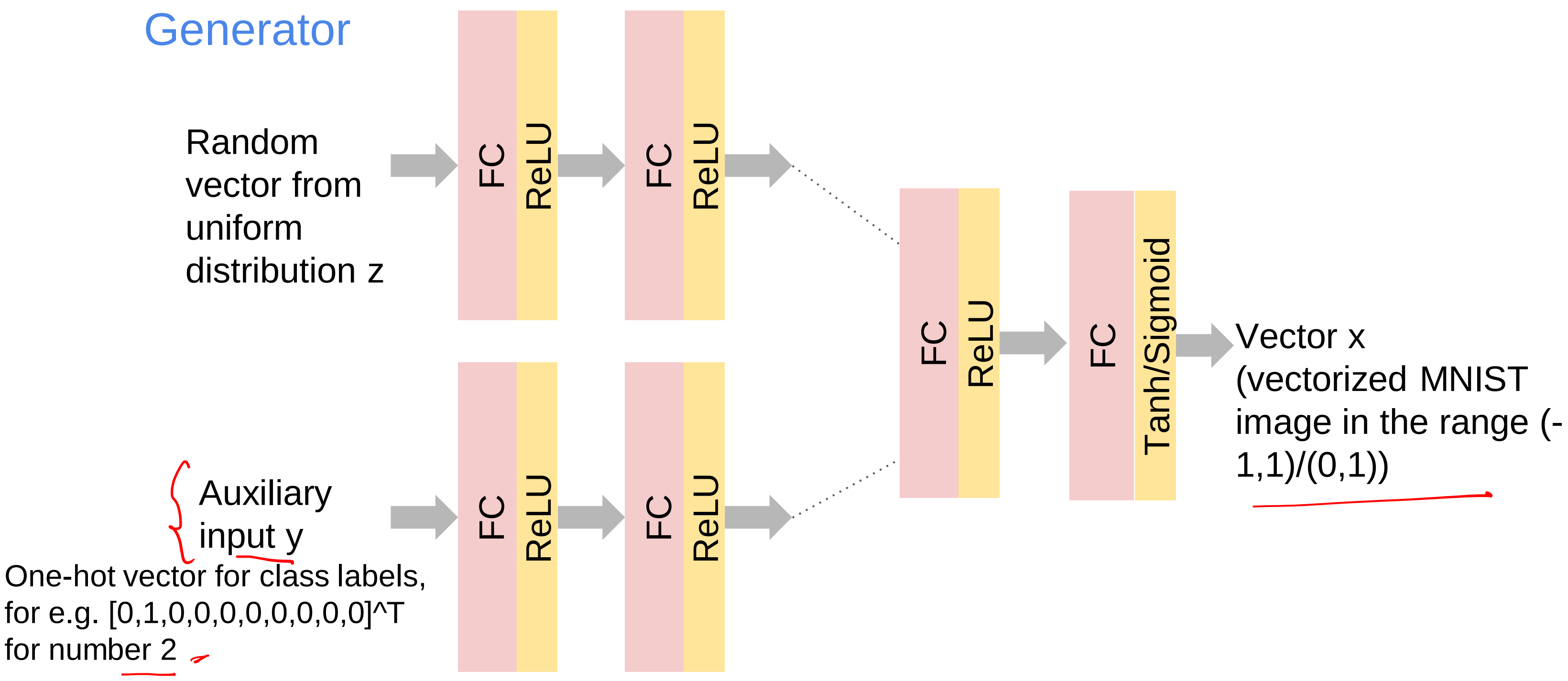
The exact layers and sizes as in Mirza and Osindero “Conditional Generative Adversarial Nets” 2014 are not shown here.

Conditional GAN - MNIST



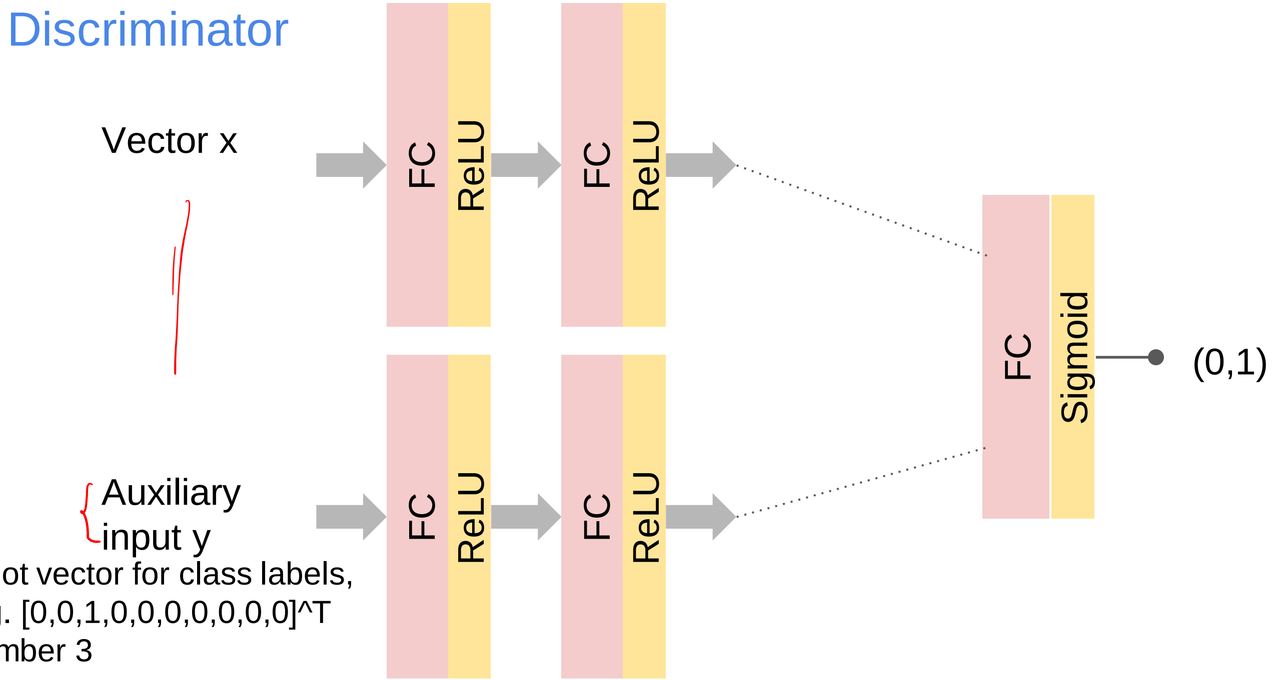
The exact layers and sizes as in Mirza and Osindero “Conditional Generative Adversarial Nets” 2014 are not shown here.

Conditional GAN - MNIST



The exact layers and sizes as in Mirza and Osindero "Conditional Generative Adversarial Nets" 2014 are not shown here.

Conditional GAN - MNIST

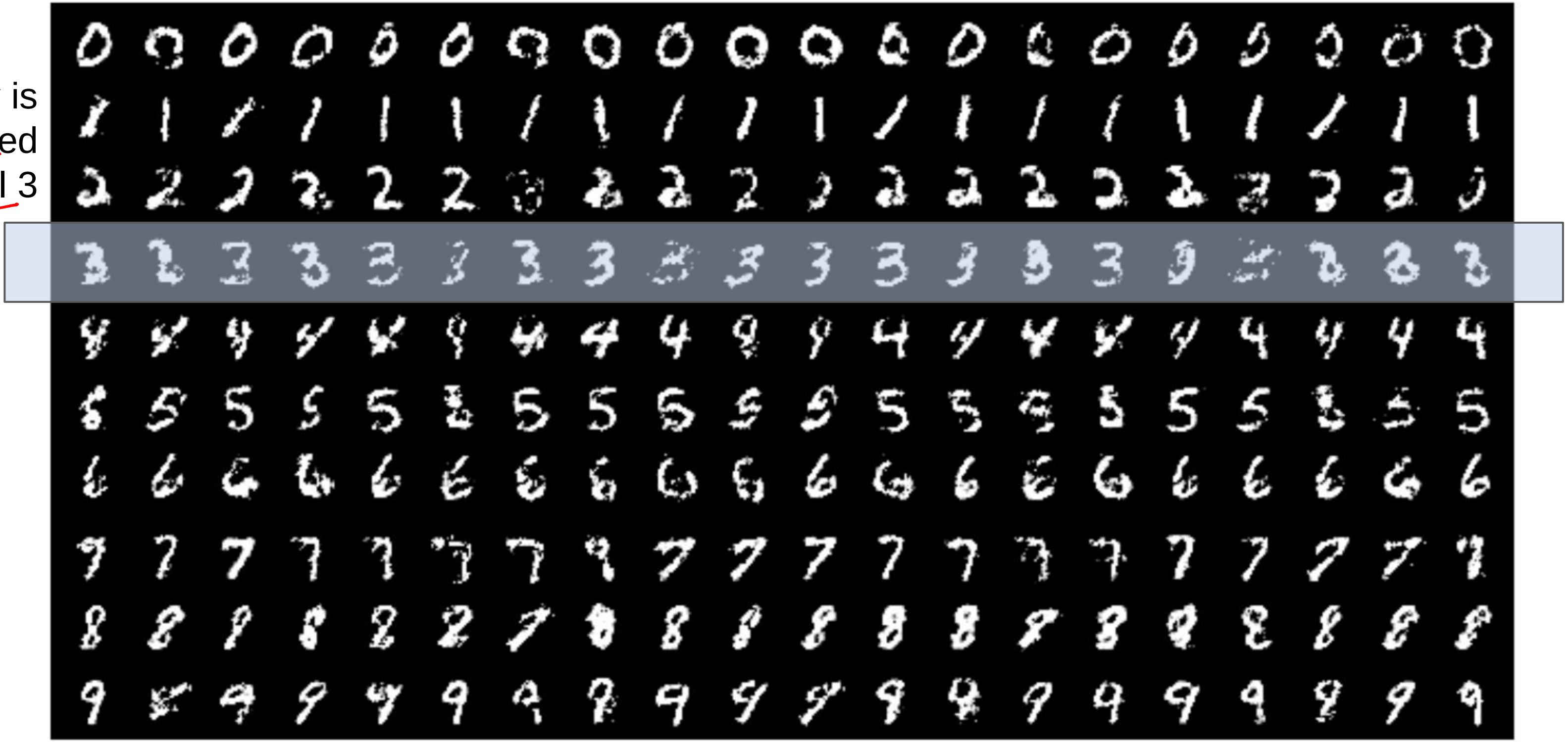


One-hot vector for class labels, for e.g. [0,0,1,0,0,0,0,0,0,0]^T for number 3

The exact layers and sizes as in Mirza and Osindero “Conditional Generative Adversarial Nets” 2014 are not shown here. The paper uses maxout units in discriminator.

Conditional GAN - MNIST

this row is
conditioned
on the label 3



Interesting Blogs

https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

https://towardsdatascience.com/turning-fortnite-into-pubg-with-deep-learning-cyclegan-2f9d339dcdb0