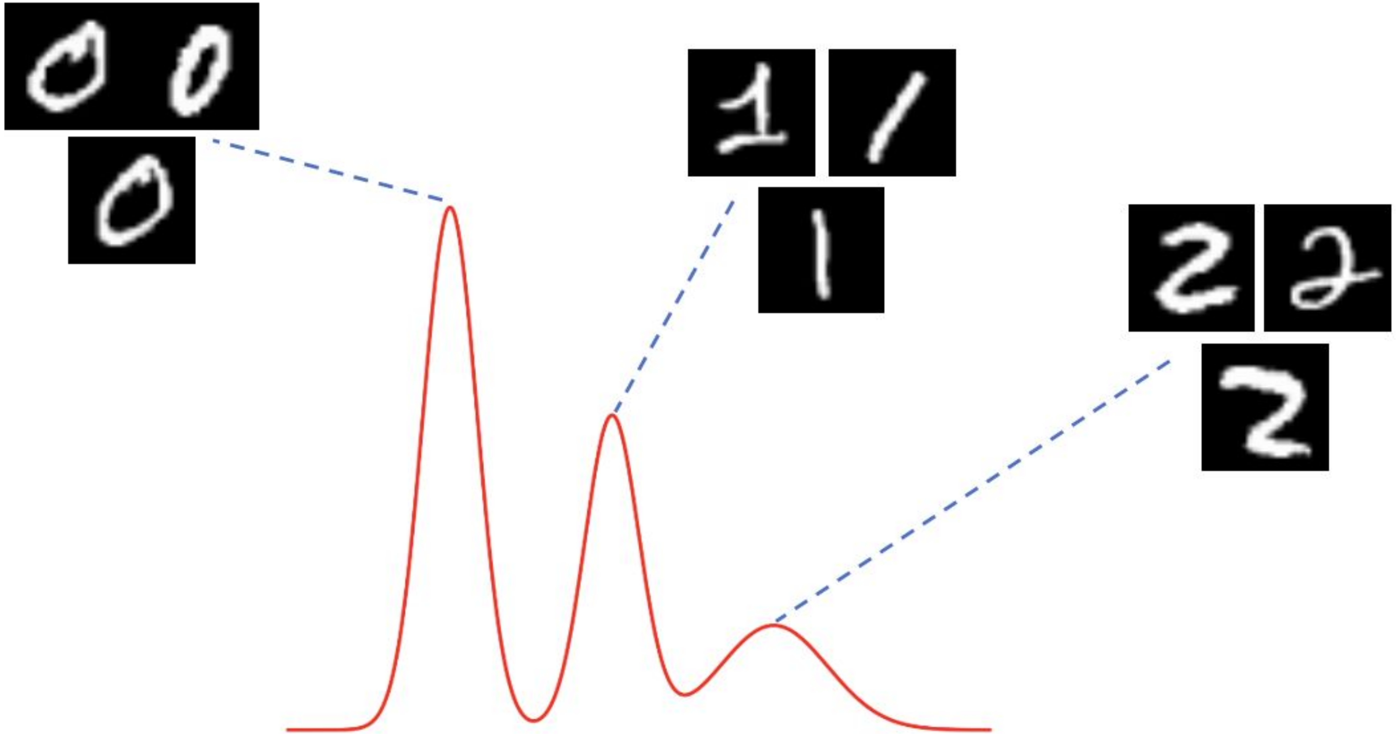
Practices in visual computing 2

Lab3: GANs

Simon Fraser University
Spring 2025

Generative learning



Generative learning

Explicit Models:

- train through directly optimizing p(x)
- Through Maximum likelihood estimation (MLE)
- Example: VAEs

Implicit Models:

- Train to sample from p(x)
- Example: GANs

GAN Conceptual Introduction

Two networks (generator G and discriminator D) compete in a minimax game.

Generator (G): Learns to produce “fake” data that mimic real data.

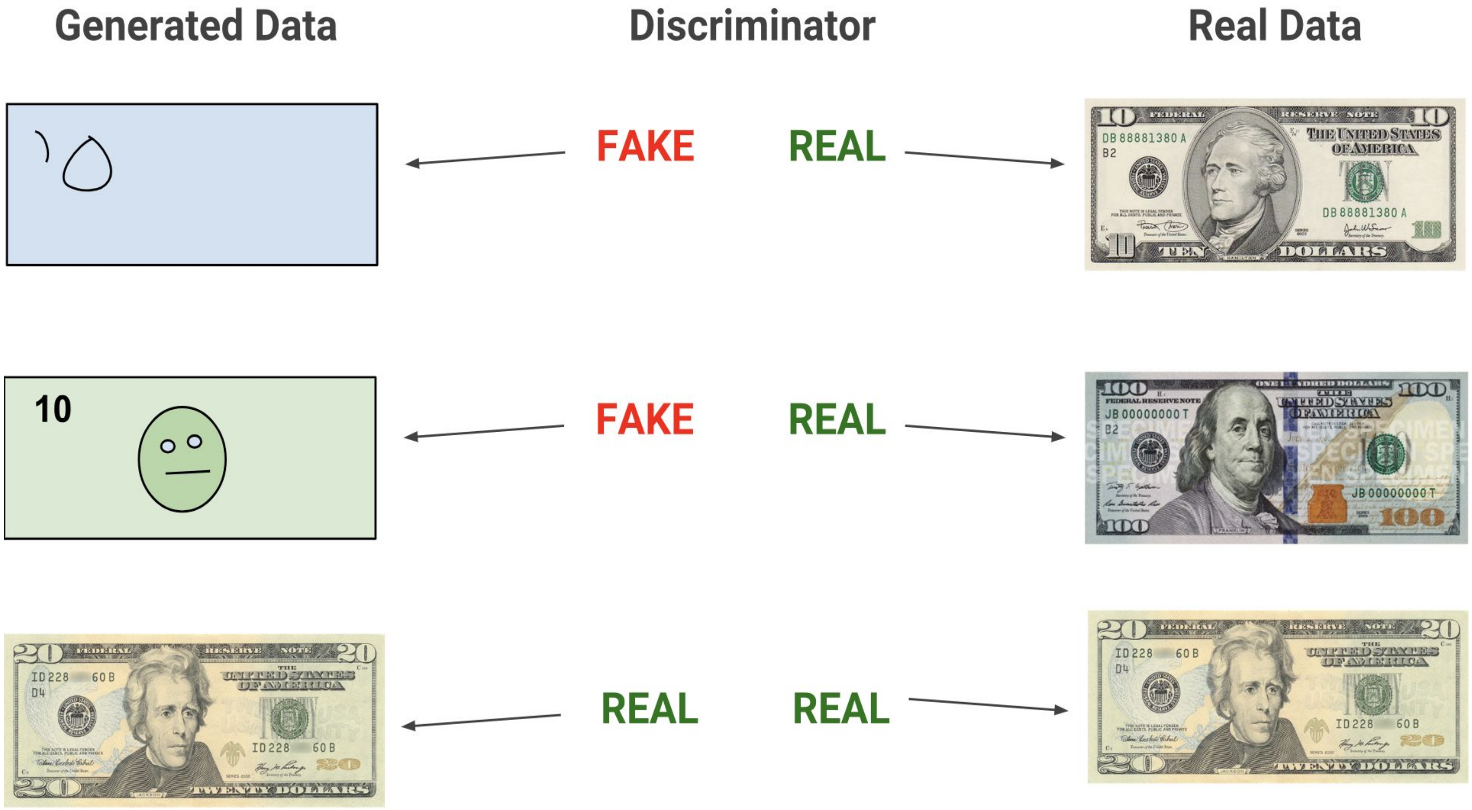
Discriminator (D): Learns to distinguish between real and fake data.

As training progresses: G gets better at fooling D.

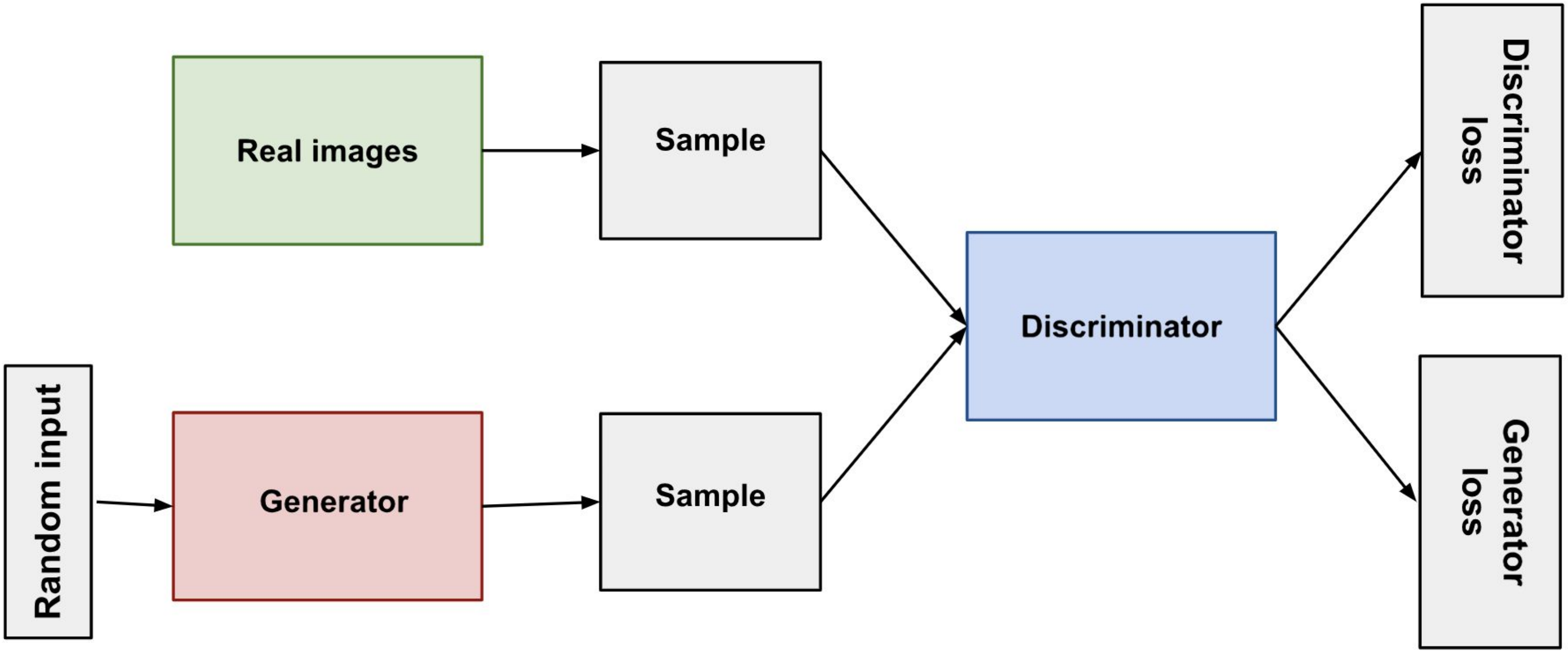
D gets better at identifying fakes.

Goal: The generator produces samples indistinguishable from real ones, and the discriminator fails to tell them apart.

Training GANs



GAN Architecture Diagram



Training GANs

Jointly Train Generator and Discriminator through a minmax loss

- Alternate between:
  - Gradient ascent for Discriminator

$$\max_{\theta_d}[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 \, - D_{\theta_d}\left(G_{\theta_g}(x)\right))]$$

  - Gradient ascent for Discriminator

$$\min[\mathbb{E}_{z\sim p_z} \log\left(1 \, - D_{\theta_d}\left(G_{\theta_g}(x)\right)\right)]$$

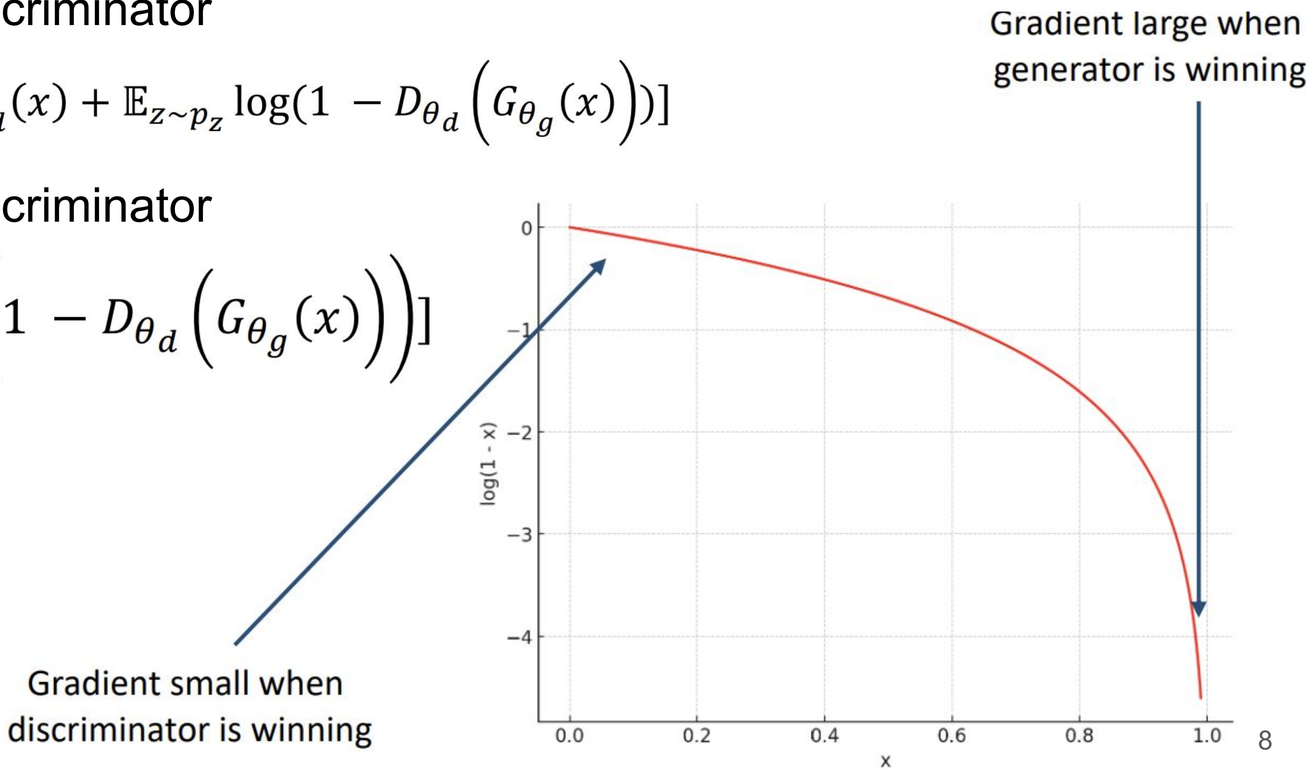
Training GANs

- Gradient ascent for Discriminator

$$\max_{\theta_d}[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}\left(G_{\theta_g}(x)\right))]$$

- Gradient ascent for Discriminator

$$\min[\mathbb{E}_{z\sim p_z} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(x)\right)\right)]$$



Training GANs

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 descending its stochastic gradient:

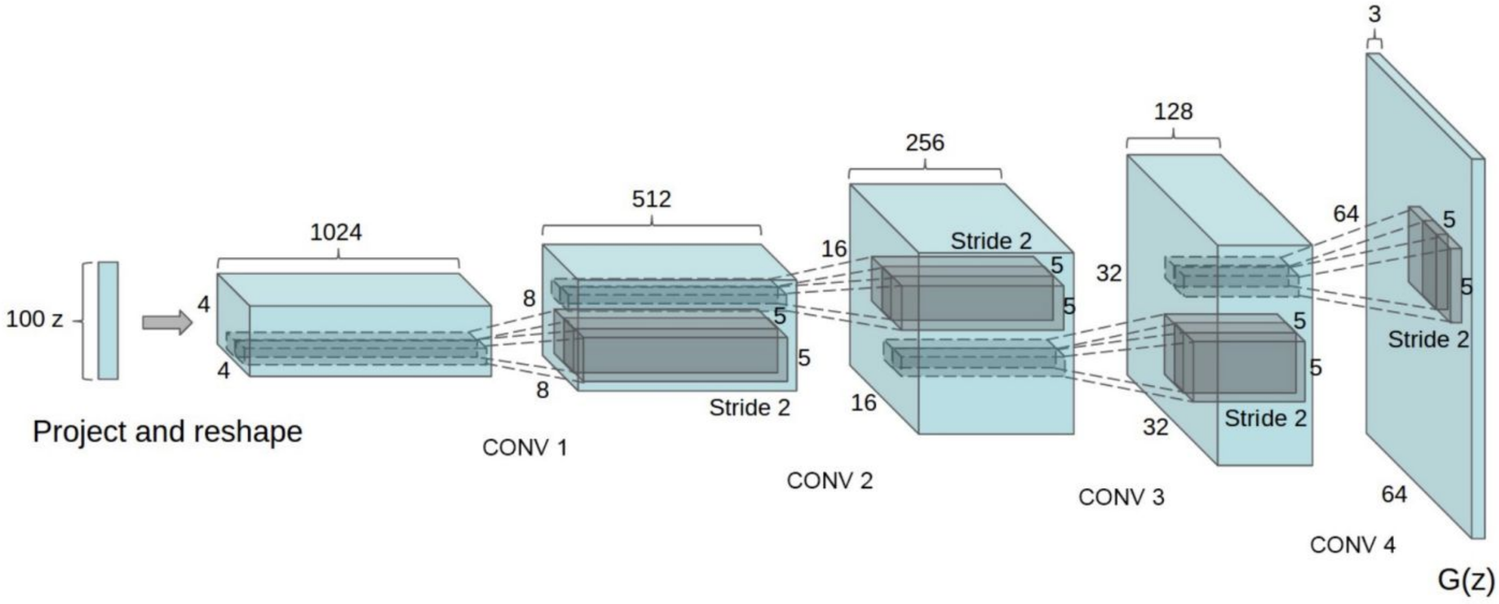
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

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

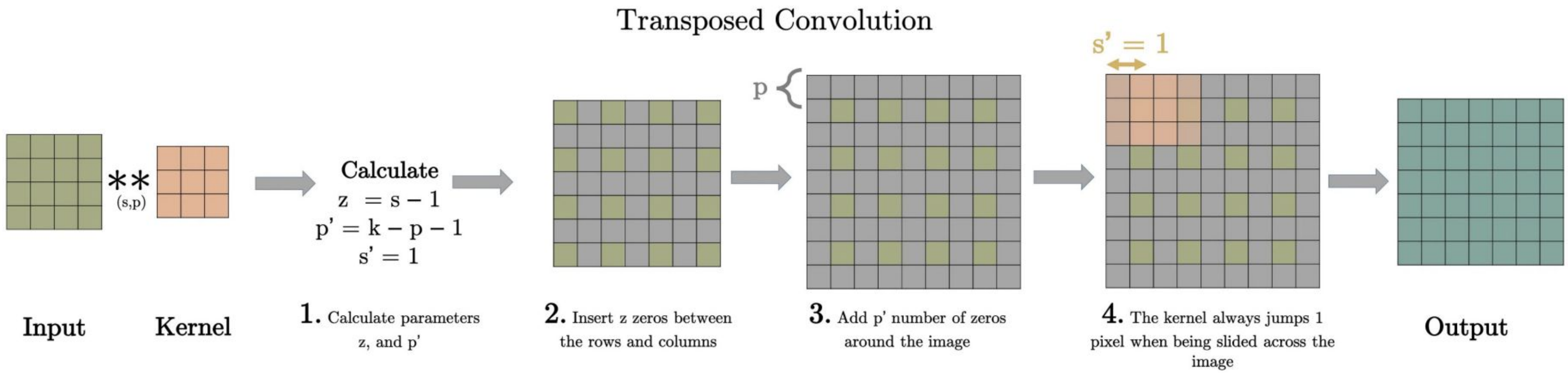
GANs: Convolutional architecture

Generator is a upsampling network with fractionally-strided Convolutions.



GANs: Convolutional architecture

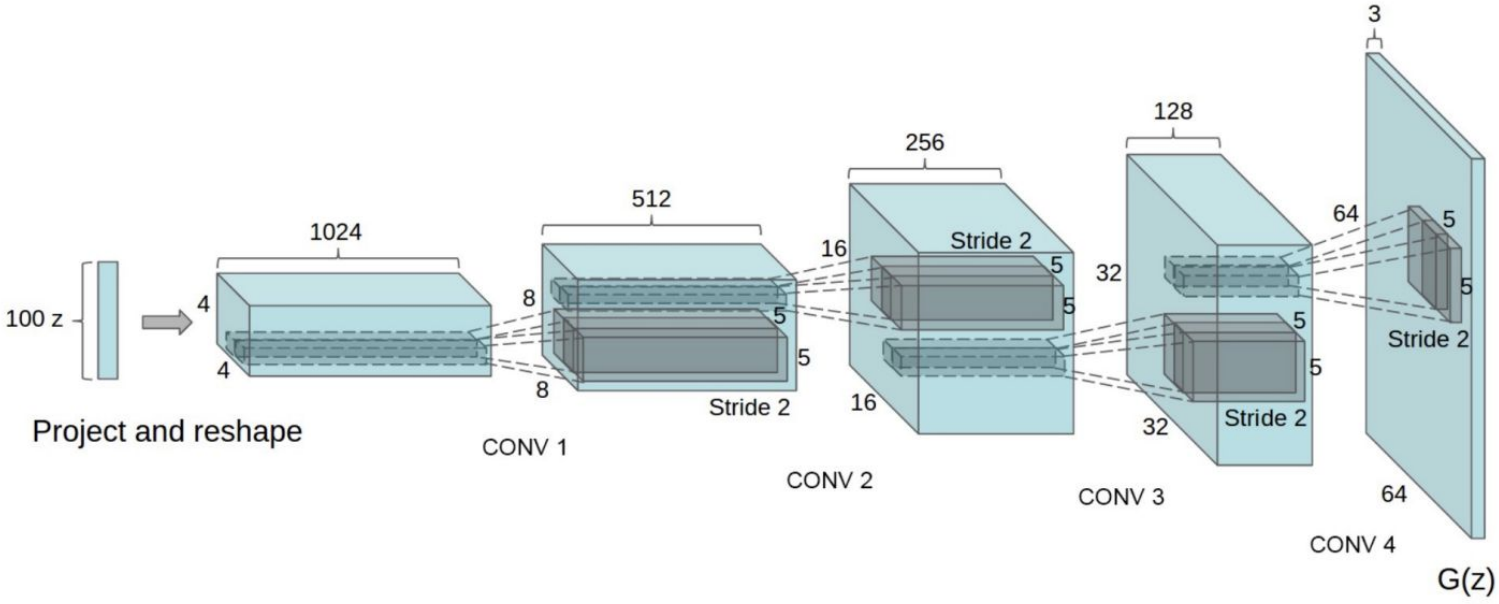
Generator is a upsampling network with fractionally-strided Convolutions.



$$o = (i - 1)\times s + k - 2p$$

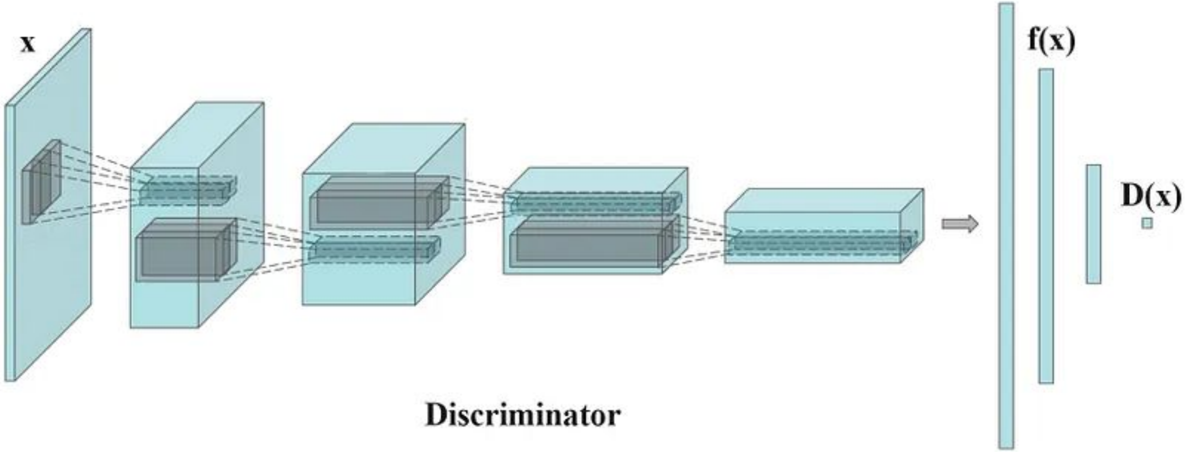
GANs: Convolutional architecture

Generator is a upsampling network with fractionally-strided Convolutions.



GANs: Convolutional architecture

- Generator is a upsampling network with fractionally-strided Convolutions.
- Discriminator is a CNN.



Tips and Tricks

Training GANs is difficult

- Stability between Generator and Discriminator
  - Vanishing Gradients - Exploding Gradients
- Mode Collapse
  - The generator repeatedly produces similar or identical samples.

Tips and Tricks

There are general tips that can help train GANs

From https://github.com/soumith/ganhacks

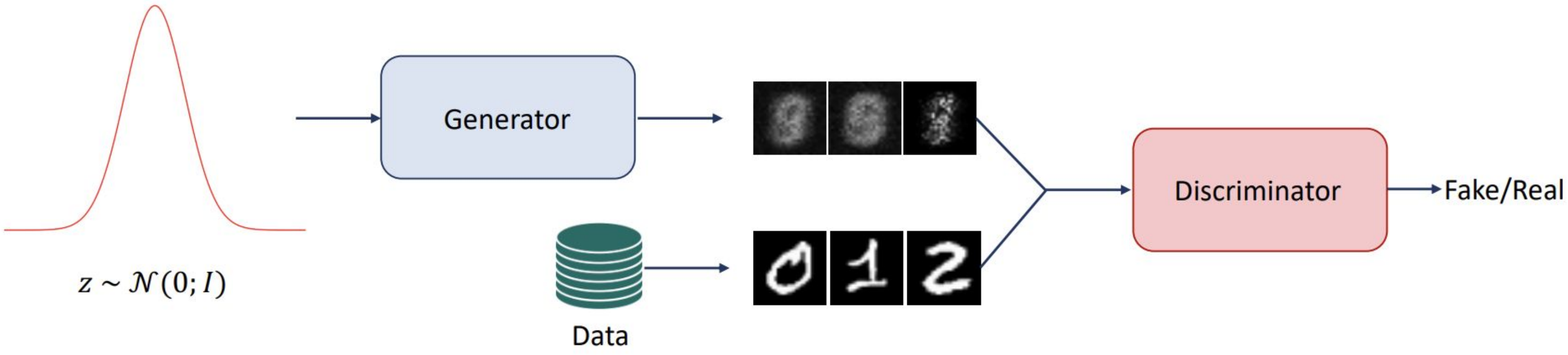
Normalize input in range (-1, 1)

- Use Tanh activation for the last layer of the Generator output

1. Avoid sparse gradients
    - Avoid using ReLU and maxpool
2. Track failures early
    - If Discriminator gradients go to zero: failure mode
    - If there is high variance/spikes in the Discriminator loss: failure mode
    - If loss of Generator steadily decreases it is fooling the Discriminator with garbage
3. Etc.

GAN progress on face generation

GAN MNIST



Reference

https://developers.google.com/machine-learning/gan/gan_structure

https://machinelearningmastery.com/how-to-code-the-generative-adversarial-network-training-algorithm-and-loss-functions/

https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b

Slides from last year of CMPT743