Good books:

- **Hands-On Generative Adversarial Networks with PyTorch 1.x**
  By John Hany and Greg Walters, published by Packt, 2019
  - Examples in PyTorch
  - Surprisingly good at theory and tips

- **GANs in Action**
  By Jakub Langr and Vladimir Bok, published by Manning, 2019
  - Examples in TensorFlow2/Keras
  - Good format – detailed explanation of the codes
Vanilla GAN

Loss Function: $L(D,G)$

$$
\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]
$$

$$
= \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D(x))]
$$

$D(x)$: probability that the real data $x$ is correctly classified by the discriminator.

$G(z)$: the fake data produced by the generator

$D(G(z))$: probability that the fake data $G(z)$ is wrongly classified by the discriminator.

$p_r(x)$: Data distribution over real sample $x$, $p_z(z)$: Data distribution over noise input $z$

$p_g(x)$: The generator’s distribution over data $x$

BCE Loss
LSGAN (Least Square GAN) – change of the loss function

\[
\begin{align*}
\min_D L(D) &= \mathbb{E}_{x \sim p_r(x)} [(1 - D(x))^2] + \mathbb{E}_{z \sim p_z(z)} [D(G(z))^2] \\
\min_G L(G) &= \mathbb{E}_{z \sim p_z(z)} [(1 - D(G(z)))^2]
\end{align*}
\]

Least Square Loss

WGAN (Wasserstein GAN) – Mathematical, later in the future

- Define a different cost/loss function from that for vanilla GANs to stabilize the training process.

PGGAN (Progressive GAN) – GPU intensive, later in the future

- Progressive increase in the number of layers
Conditional GAN (CGAN)
InfoGAN

Generator

Discriminator
StackGAN → StackGAN++ (text description → image)