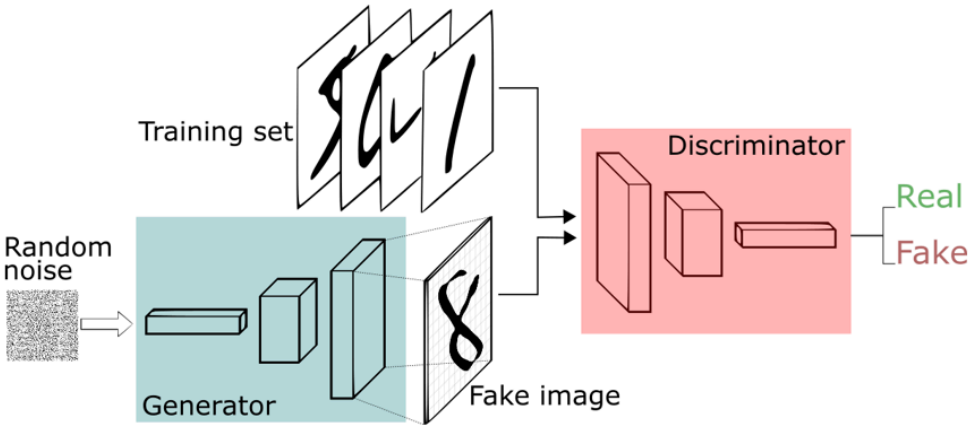
Generative Adversarial Networks (GANs) Explanation

1. What is a Generative Adversarial Network (GAN)?

Generative Adversarial Networks (GANs) were introduced by Ian Goodfellow et al. in 2014. GANs consist of two neural networks that are trained in opposition to each other to generate realistic data. These two networks are:

- **Generator**: The network that generates data from random noise.
- **Discriminator**: The network that classifies whether the data is real (from the true data distribution) or fake (generated by the generator).



2. Basic Principles of GANs

The core idea of GANs is to use a **zero-sum game** where the generator and discriminator improve by continuously challenging each other:

- The **generator** tries to create data that is indistinguishable from real data.
- The **discriminator** tries to correctly classify real and fake data.

Throughout training, the generator aims to "fool" the discriminator, while the discriminator aims to correctly identify fake data. This process is akin to a "cat-and-mouse" game.

3. GAN Training Process

The training process of GANs can be described in the following two steps:

1. **Discriminator Training**: The discriminator is trained on both real data and fake data generated by the generator, and learns to distinguish between the two.
2. **Generator Training**: The generator's parameters are adjusted through backpropagation to create data that can fool the discriminator.

The loss functions can be expressed as:

- The **generator's goal** is to minimize the loss:

$$\mathcal{L}_G = \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$$

  where z is random noise, $G(z)$ is the output of the generator, and $D(G(z))$ is the discriminator's prediction on the generated data. What's mathematical meaning?

$$
\begin{aligned}
&\min \mathbb{E}_{z\sim p} \log(1-D(G(z))) \\
\Longrightarrow &\min \mathbb{E}_{z\sim p} \left[-\log(D(G(z)))\right] \\
\Longrightarrow &\min \mathbb{E}_{z\sim p} -\left[1\cdot\log(D(G(z))) + (1-1)\cdot\log(1-D(G(z)))\right] \\
\Longrightarrow &\min \mathbb{E}_{z\sim p} \operatorname{BCE}(D(G(z)), 1)
\end{aligned}
$$

- The **discriminator's goal** is to maximize the loss:

$$
\mathcal{L}_D = \mathbb{E}_{x\sim p_{data}} \log(D(x)) + \mathbb{E}_{z\sim p} \log(1-D(G(z)))
$$

where x is real data, $G(z)$ is generated data, and $D(x)$ is the discriminator's prediction on real data. What's mathematical meaning?

$$
\begin{aligned}
&\max \mathbb{E}_{x\sim p_{data}} \log(D(x)) + \mathbb{E}_{z\sim p} \log(1-D(G(z))) \\
\Longrightarrow &\min \mathbb{E}_{x\sim p_{data}} -\log(D(x)) + \mathbb{E}_{z\sim p} -\log(1-D(G(z))) \\
\Longrightarrow &\min \mathbb{E}_{x\sim p_{data}} \operatorname{BCE}(D(x), 1) + \mathbb{E}_{z\sim p} \operatorname{BCE}(D(G(z)), 0)
\end{aligned}
$$

- Procedure
  - **Step0**: Initiate classification labels for each sample for every batch. Need both **label_true** and **label_fake**
  - **Step1**: Train the discriminator by two pairs of input & label: real image with label_true, fake image with label_fake. Then using the loss function to do the backward propagation.
  - **Step2**: Train the generator by fake image with label_fake. Then using the loss function to do the backward propagation.

  Pay attention, in Step1, training of discriminator should be a isolated process, so think about how to keep the generator away from backpropagation.