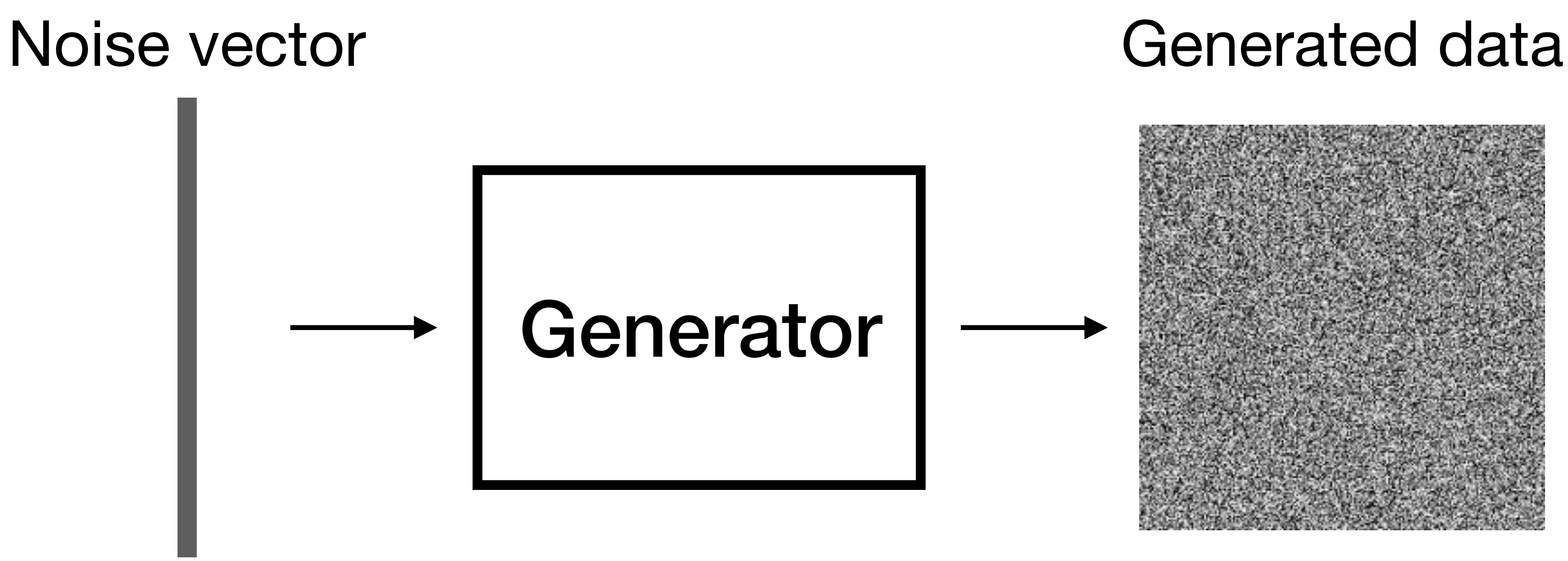
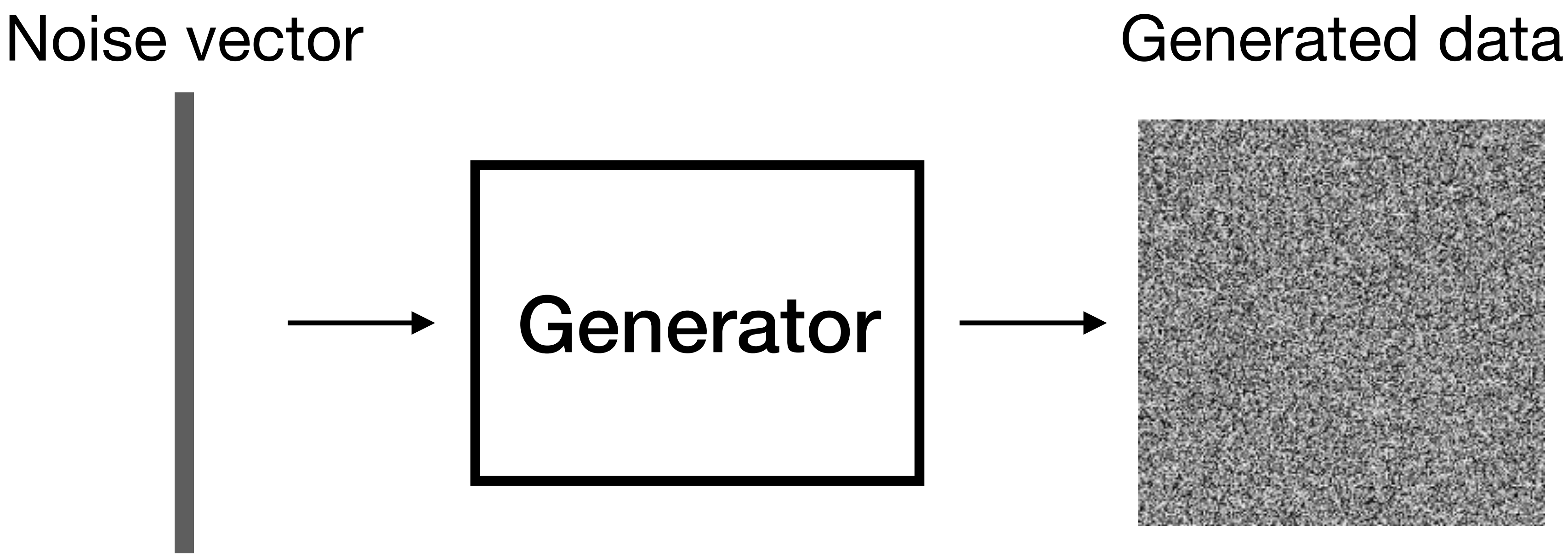
GAN

- Generative adversarial network
- Generative model
- Map a latent code to high dimension data
- Training in an adversarial way

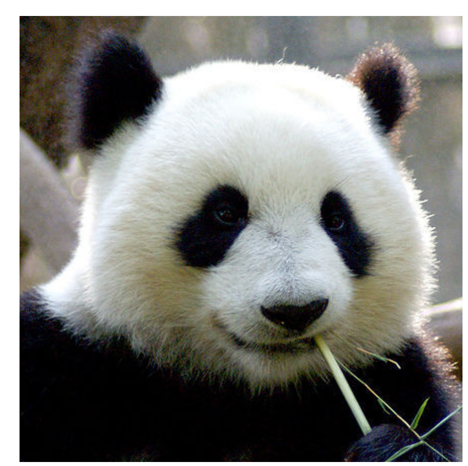
GAN Architecture



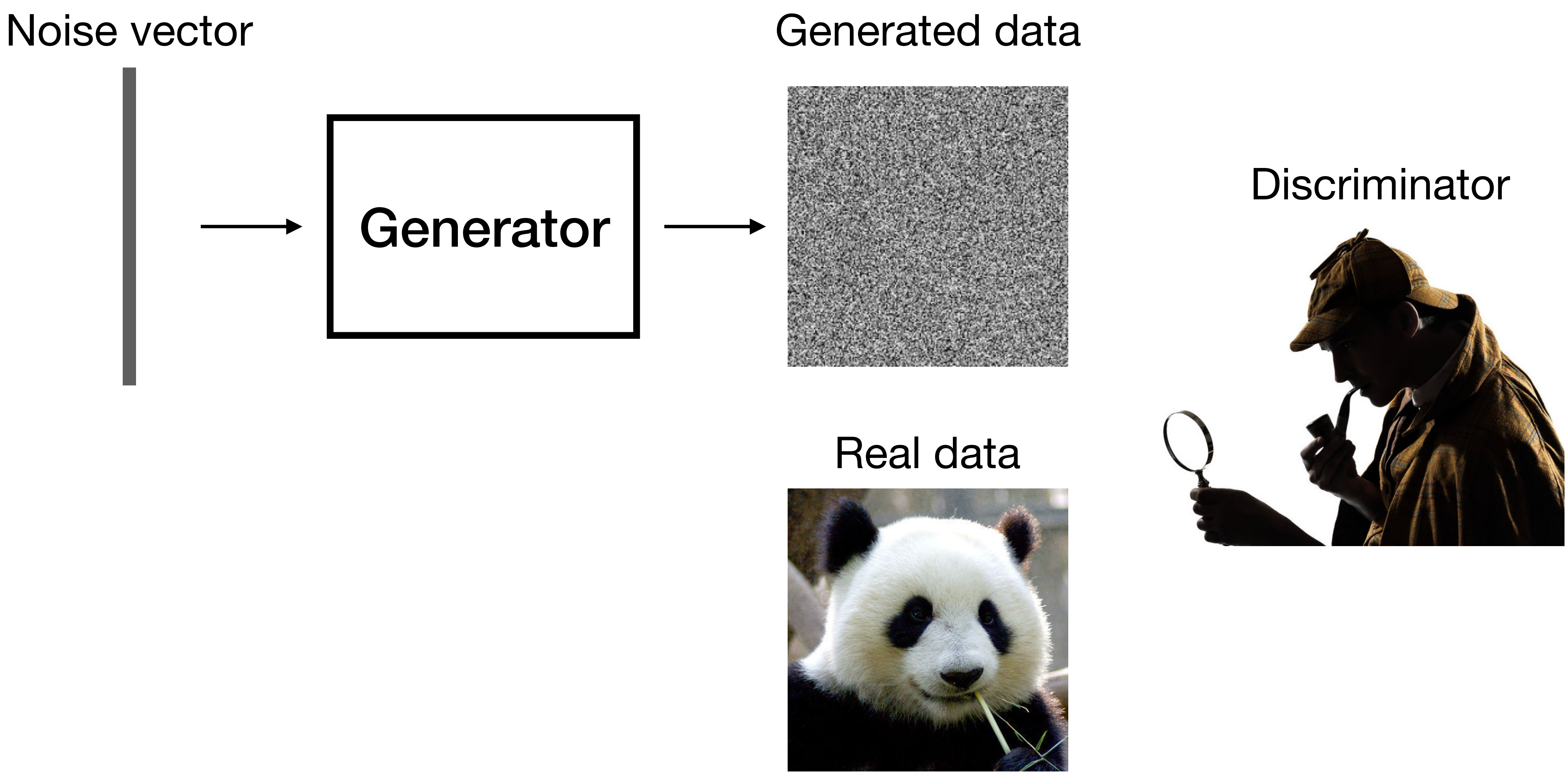
GAN Architecture



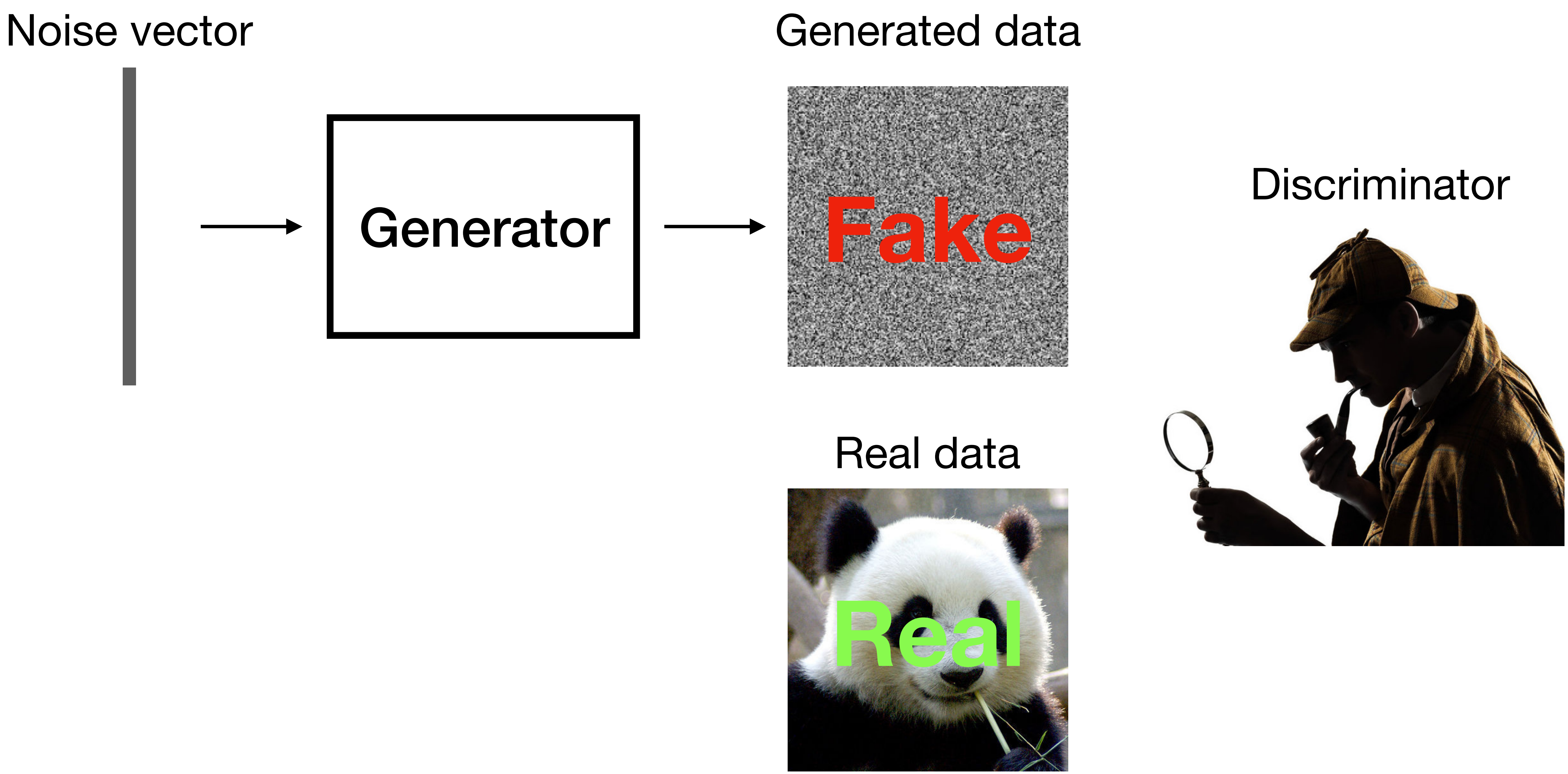
Real data



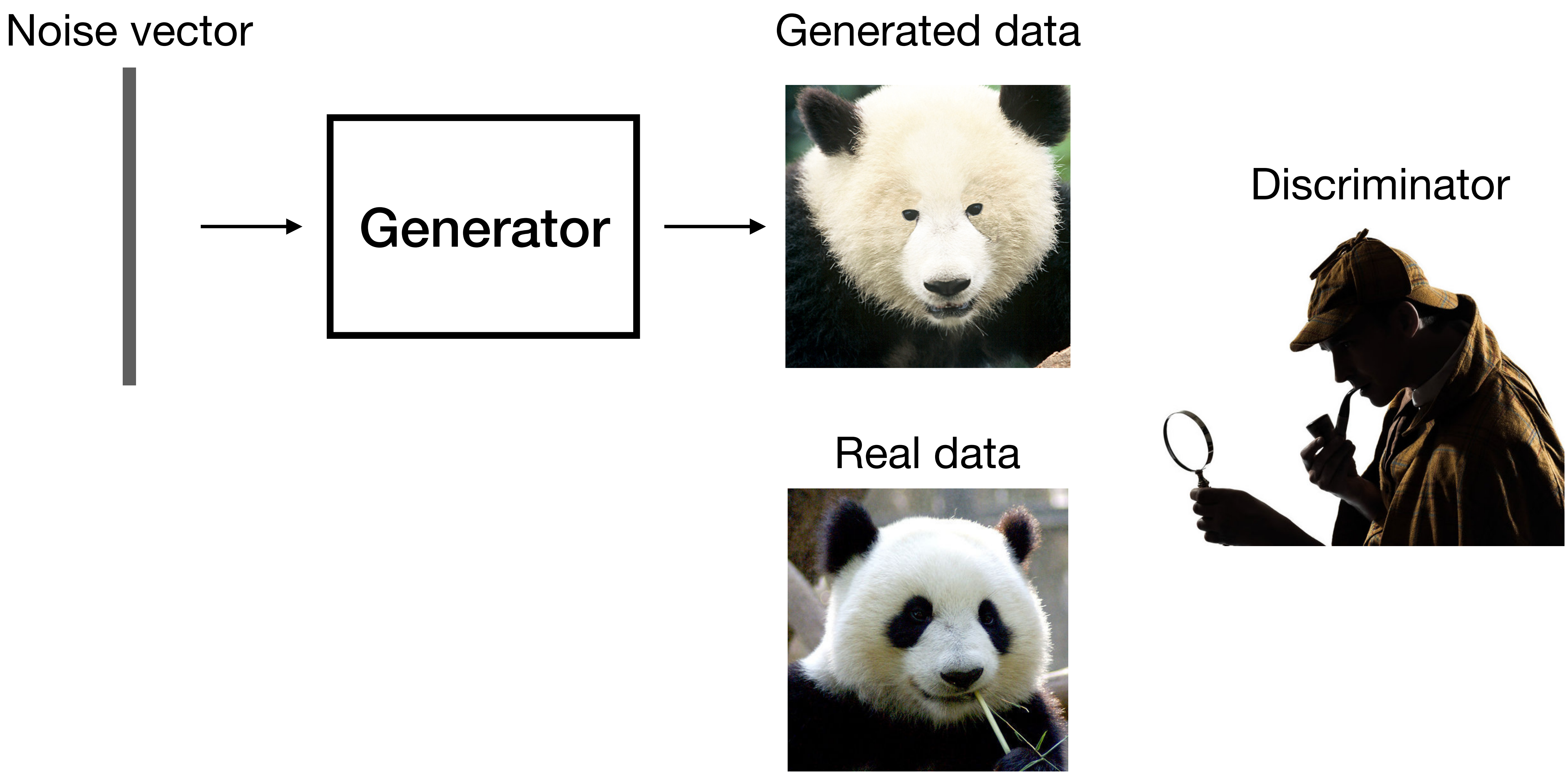
GAN Architecture



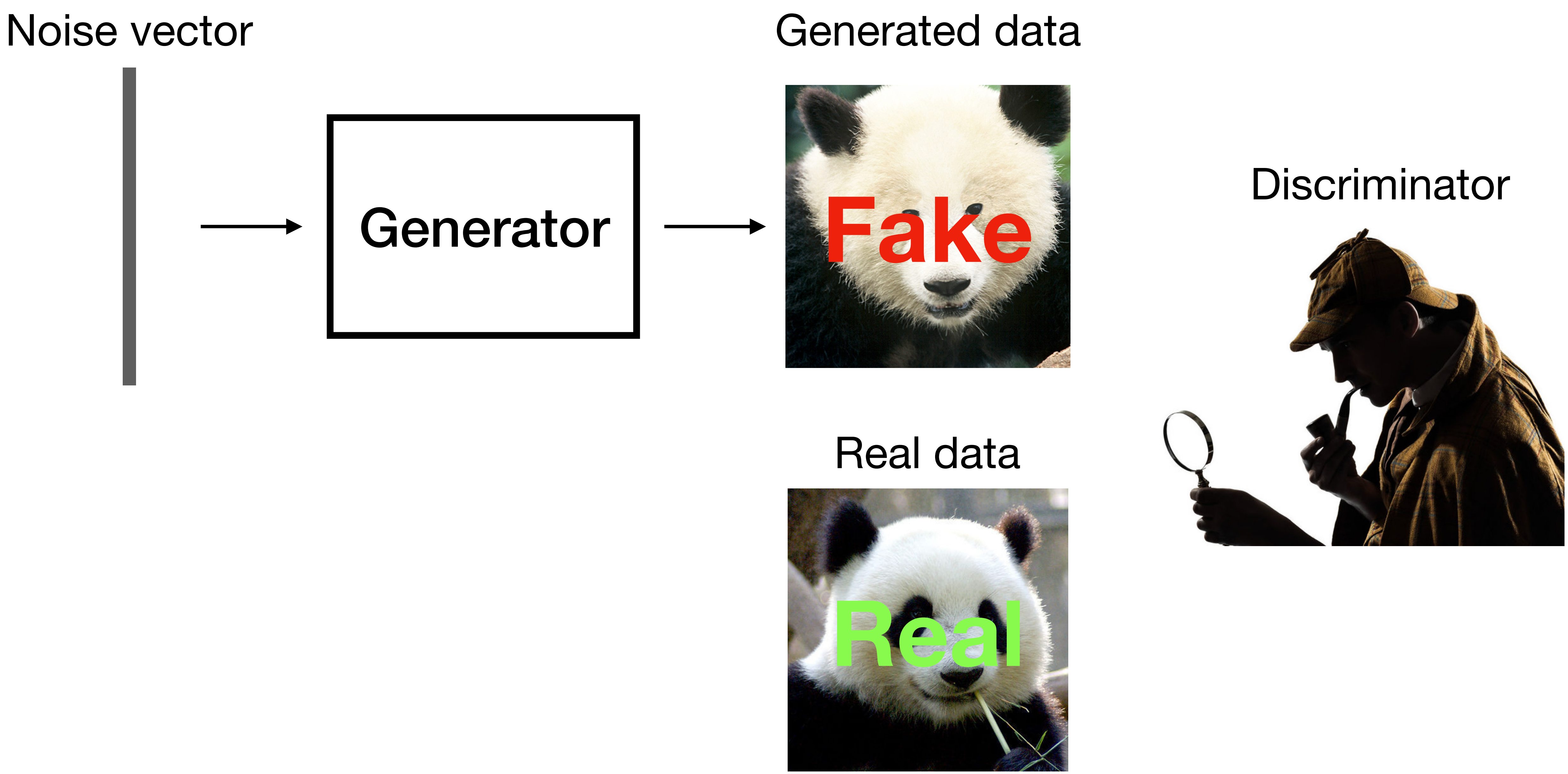
GAN Architecture



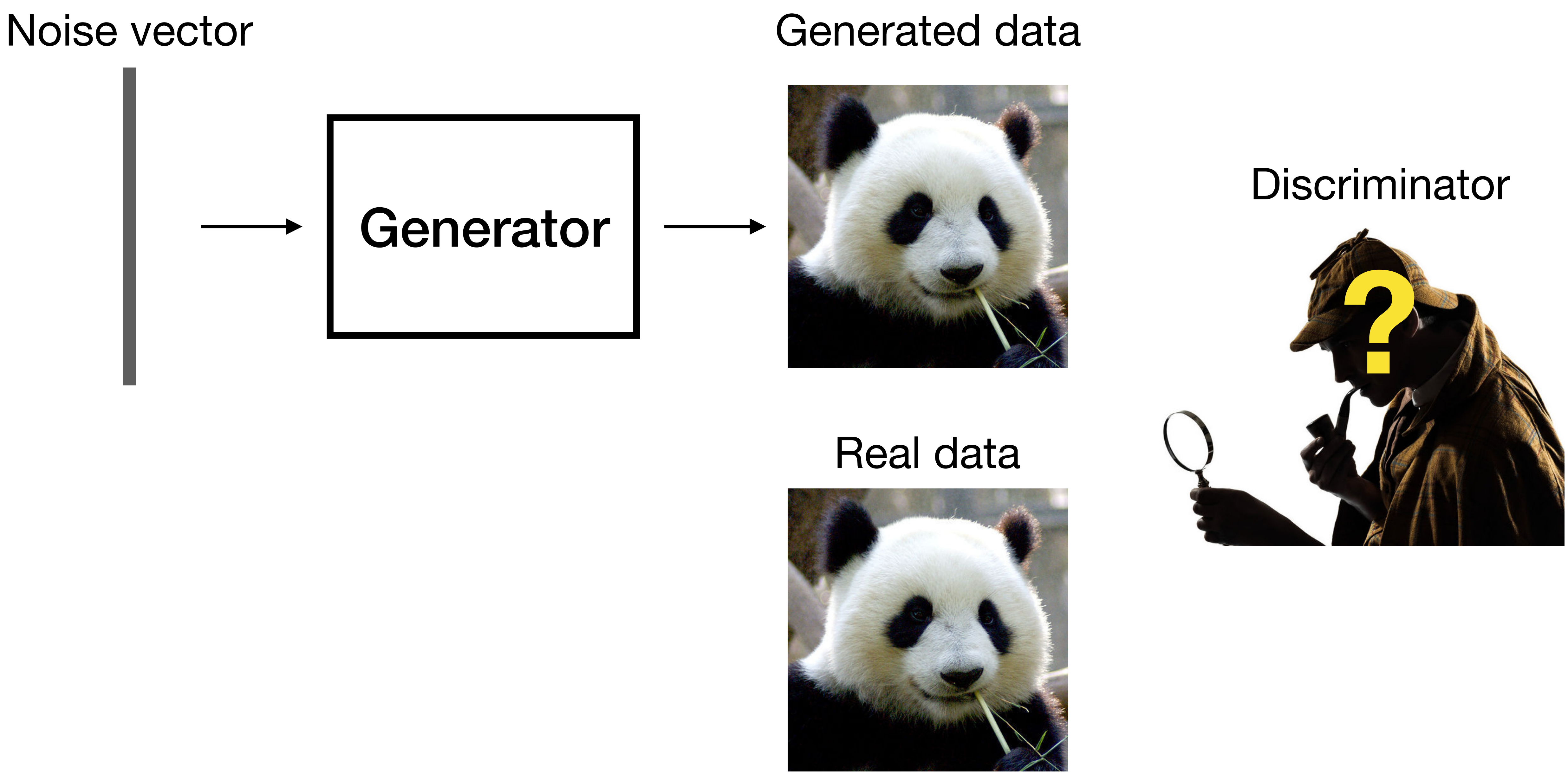
GAN Architecture



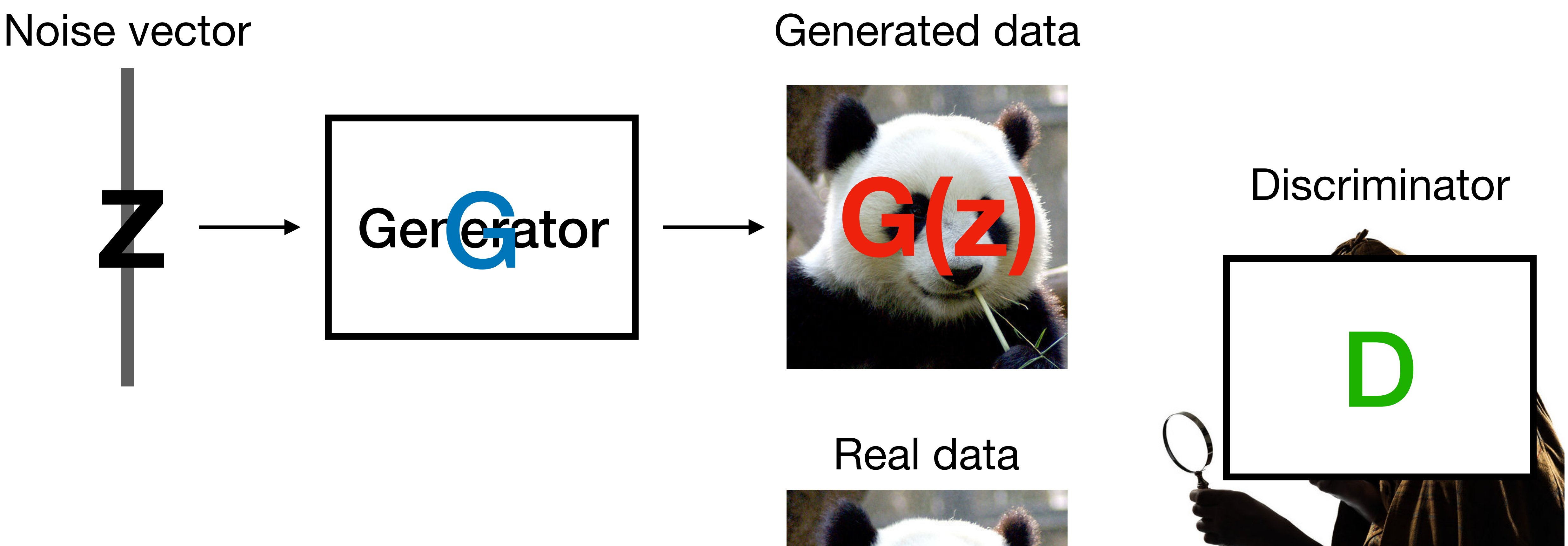
GAN Architecture

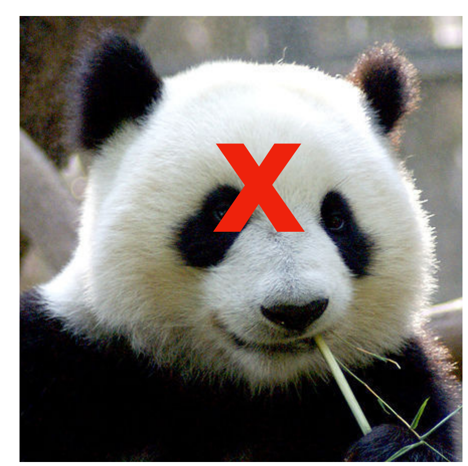


GAN Architecture



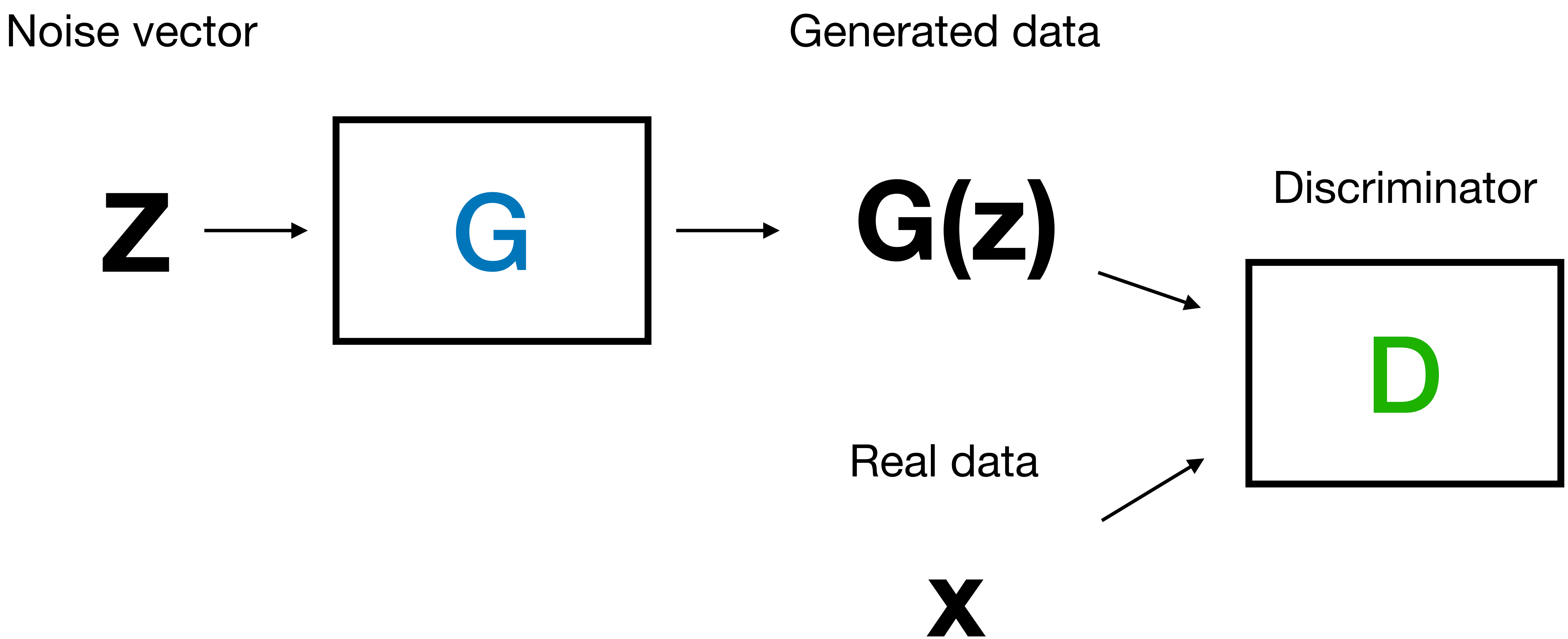
GAN Architecture





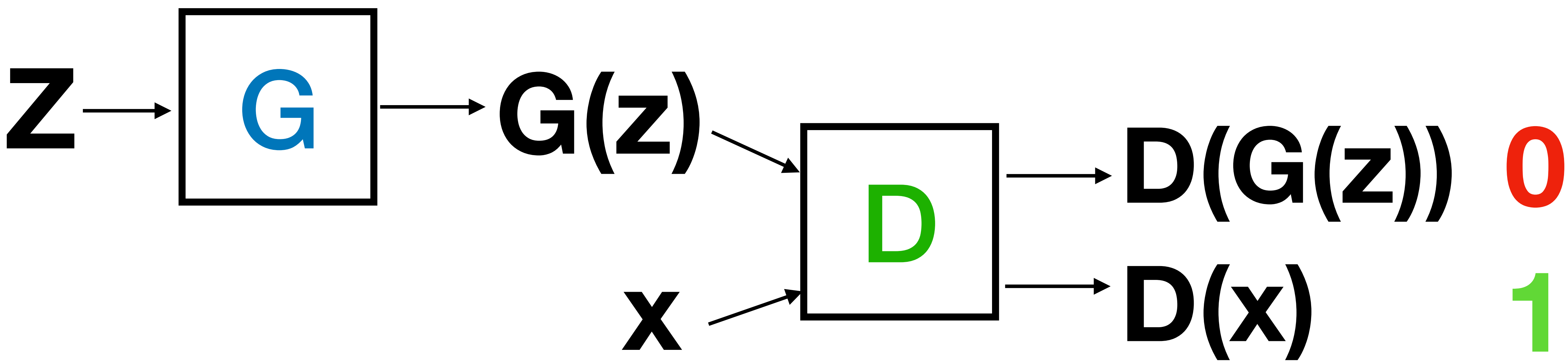
- Noise vector z
- Generator G maps from noise z to data space G(z)
- Discriminator D distinguishes G(z) and real data x

GAN Architecture



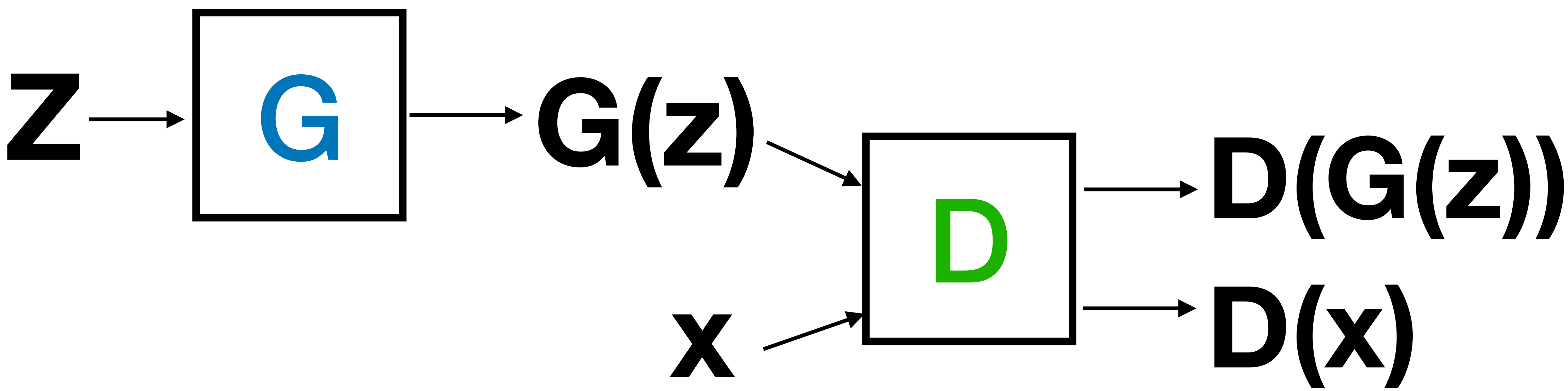
- Noise vector z
- Generator G maps from noise z to data space G(z)
- Discriminator D distinguishes G(z) and real data x

Minimax Game



- Train discriminator D to **maximise** the probability of assigning the correct label to both training samples x (label as 1, i.e. D(x):1) and samples from G (label as 0, i.e. D(G(z)):0)
- Train generator G to **minimise** the distance between the distribution of generated samples G(z) and the distribution of training samples x

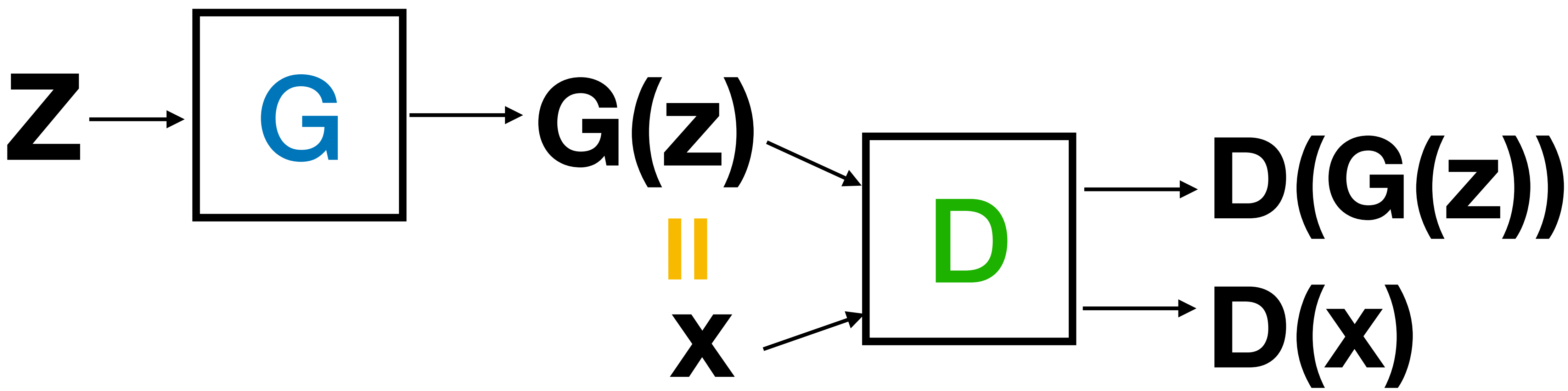
Minimax Game



Full objective:

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

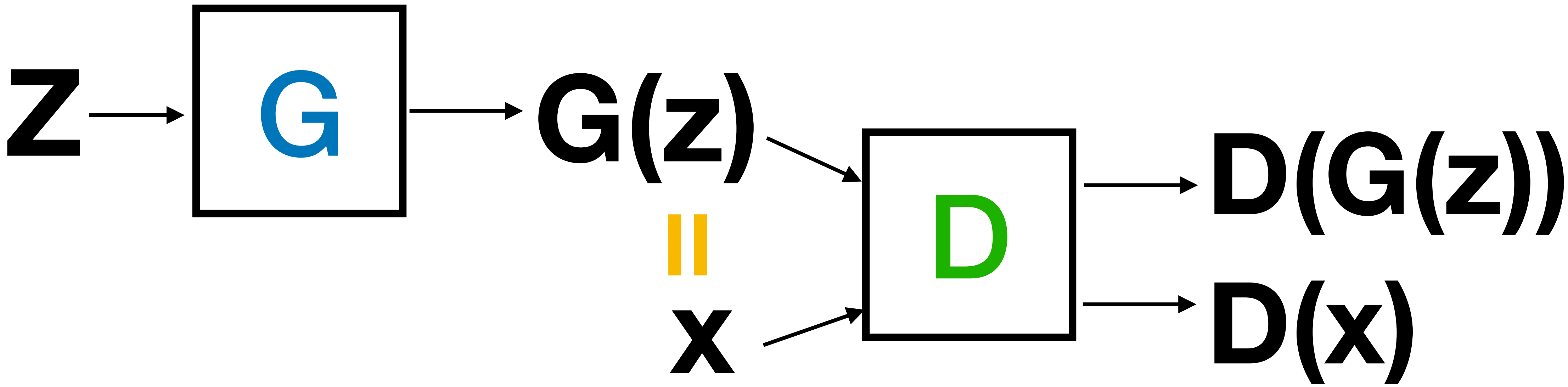
Global optimum



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

- For G, $p_g = p_{data}$
- For D, it outputs $D*(x) = D*(G(z))$

Loss function

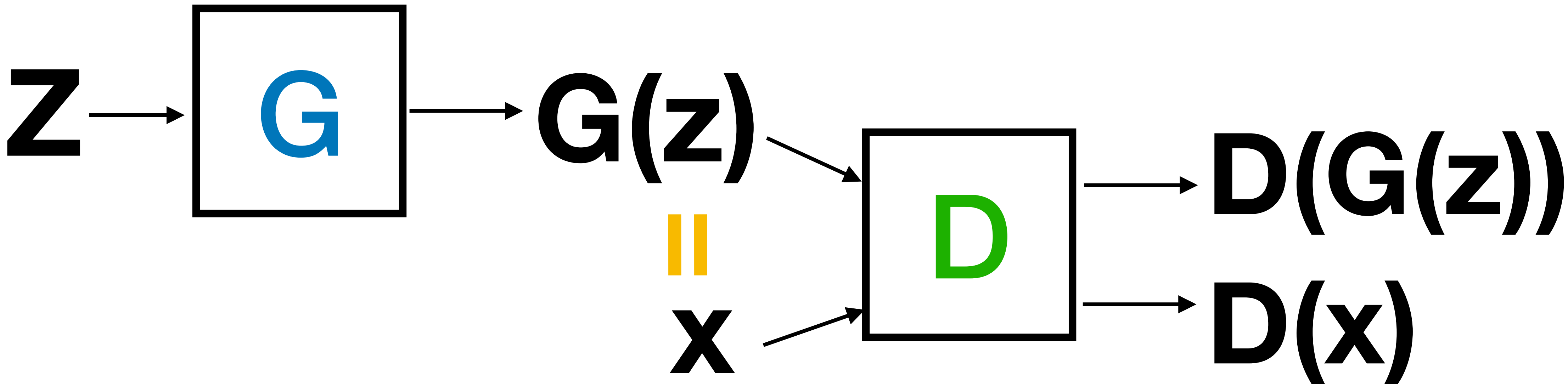


$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

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

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

Loss function



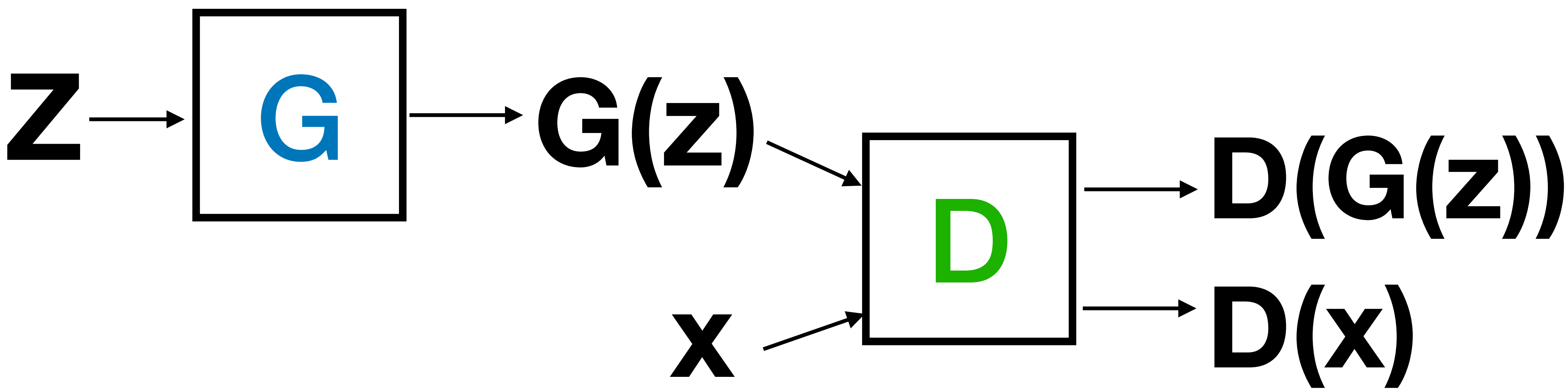
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

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

`tf.nn.sigmoid_cross_entropy_with_logits`

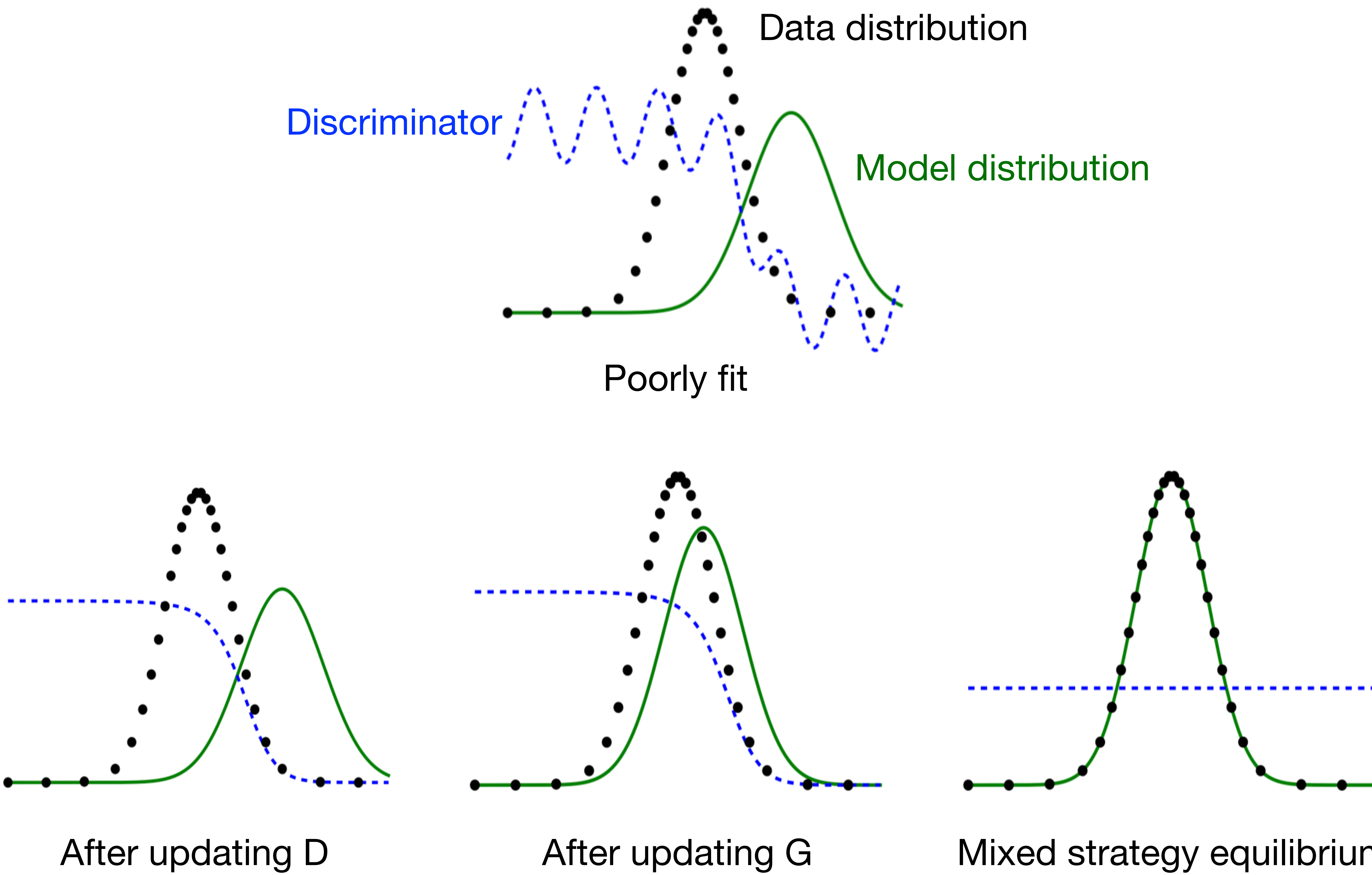
- For D, $\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$

Training Procedure

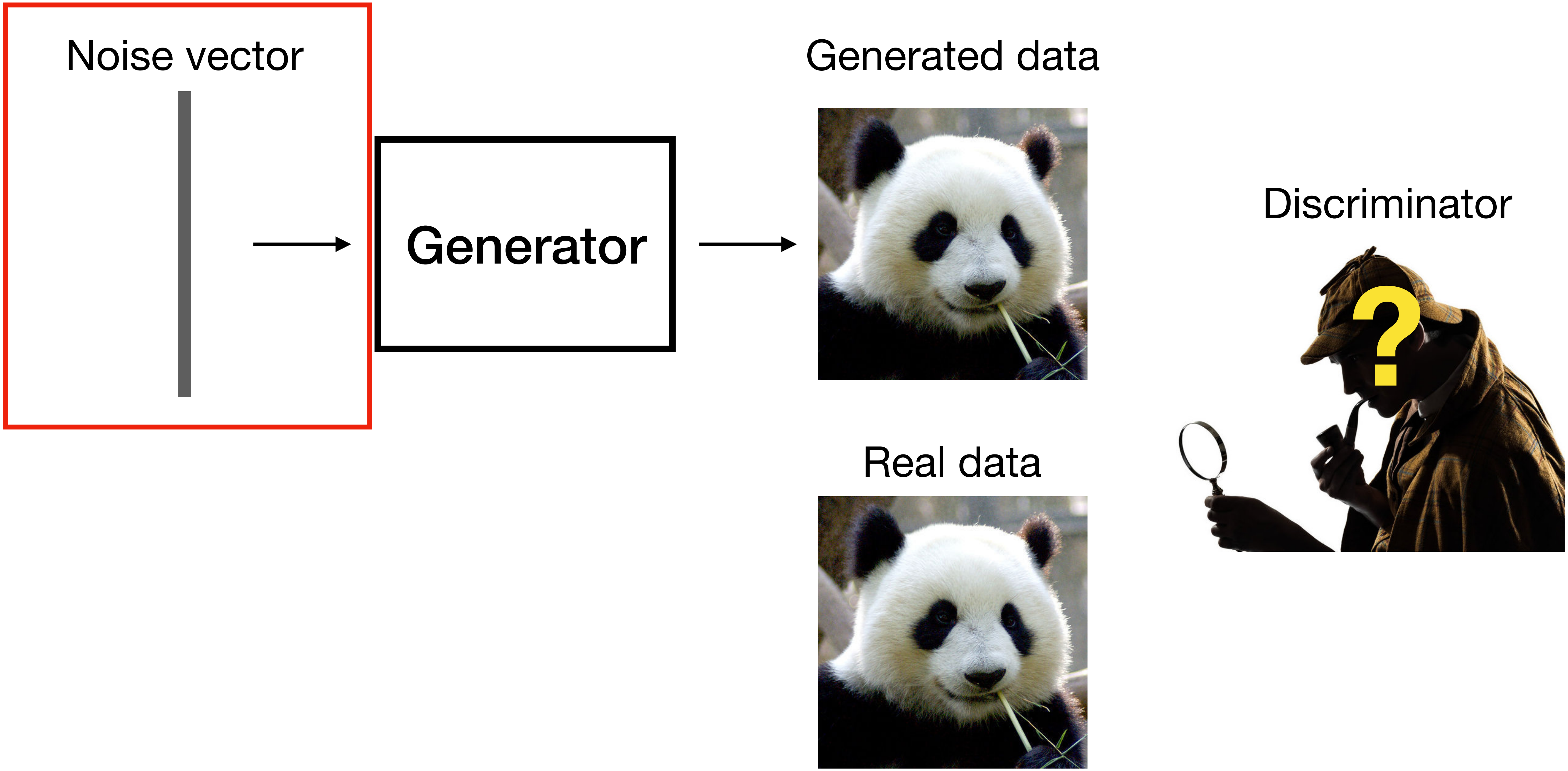


- Use SGD (Stochastic gradient descent) or Adam optimiser on generator and discriminator
- Alternately update G and D until converge

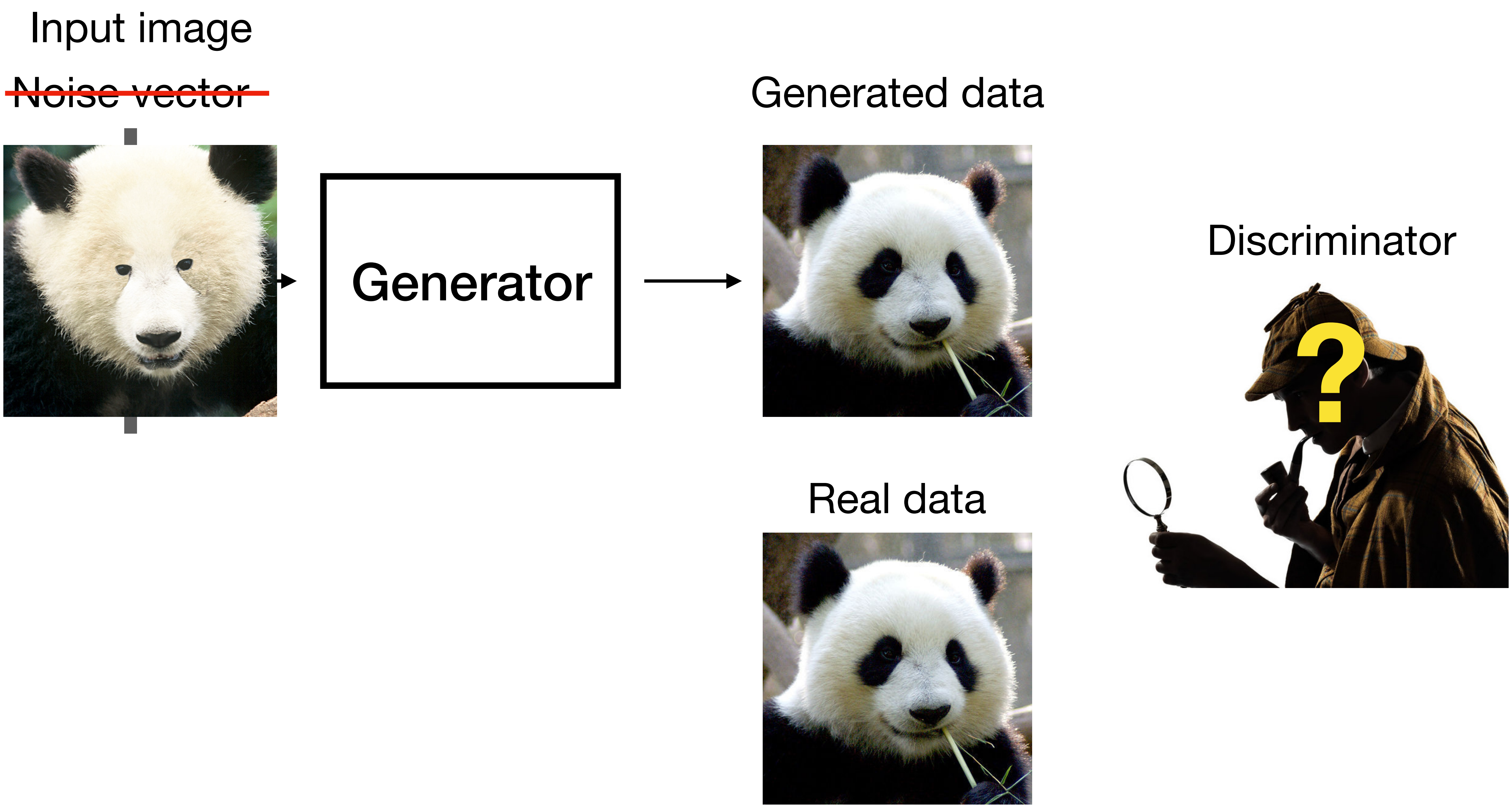
GAN


Data distribution
Discriminator
Model distribution
Poorly fit
After updating D
After updating G
Mixed strategy equilibrium

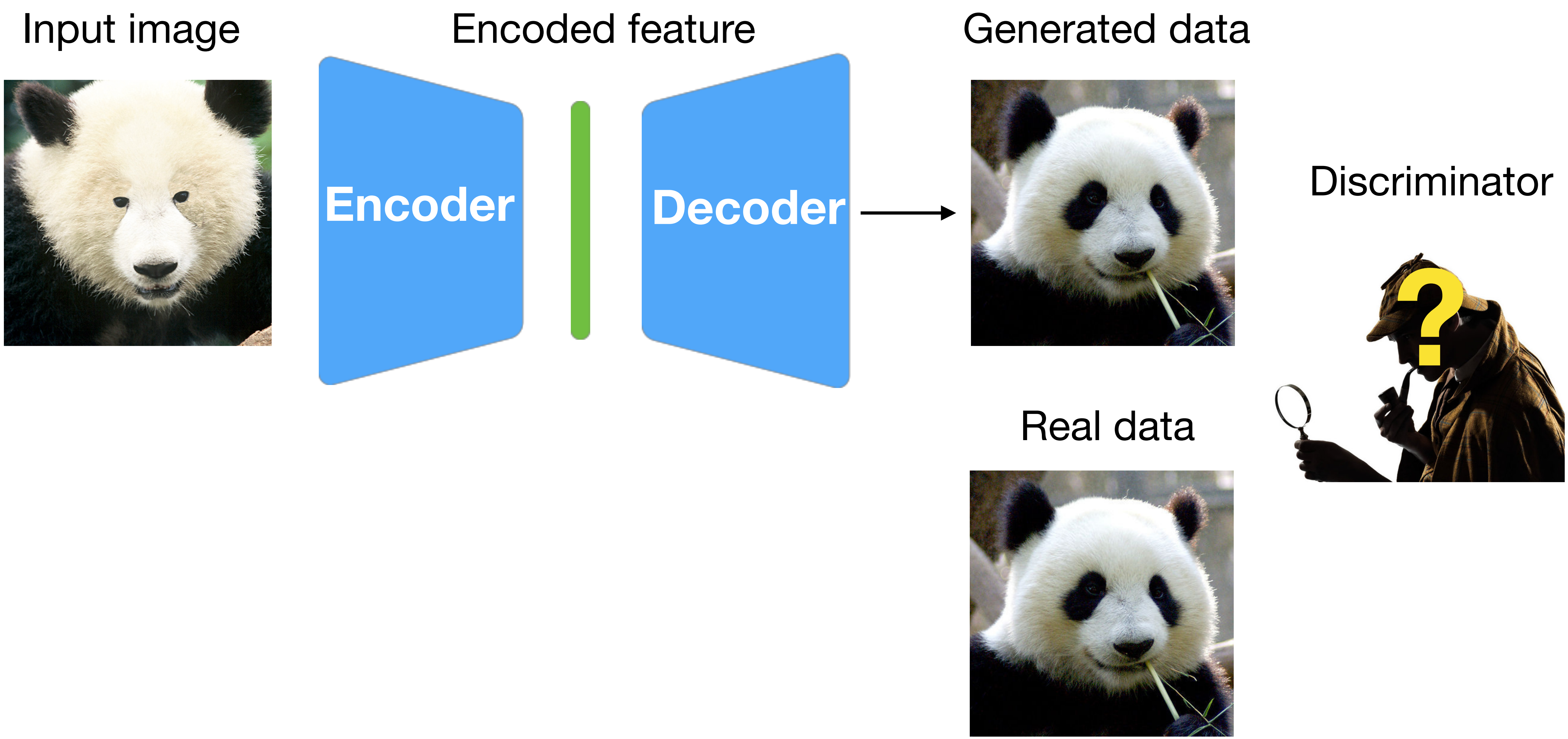
GAN Architecture



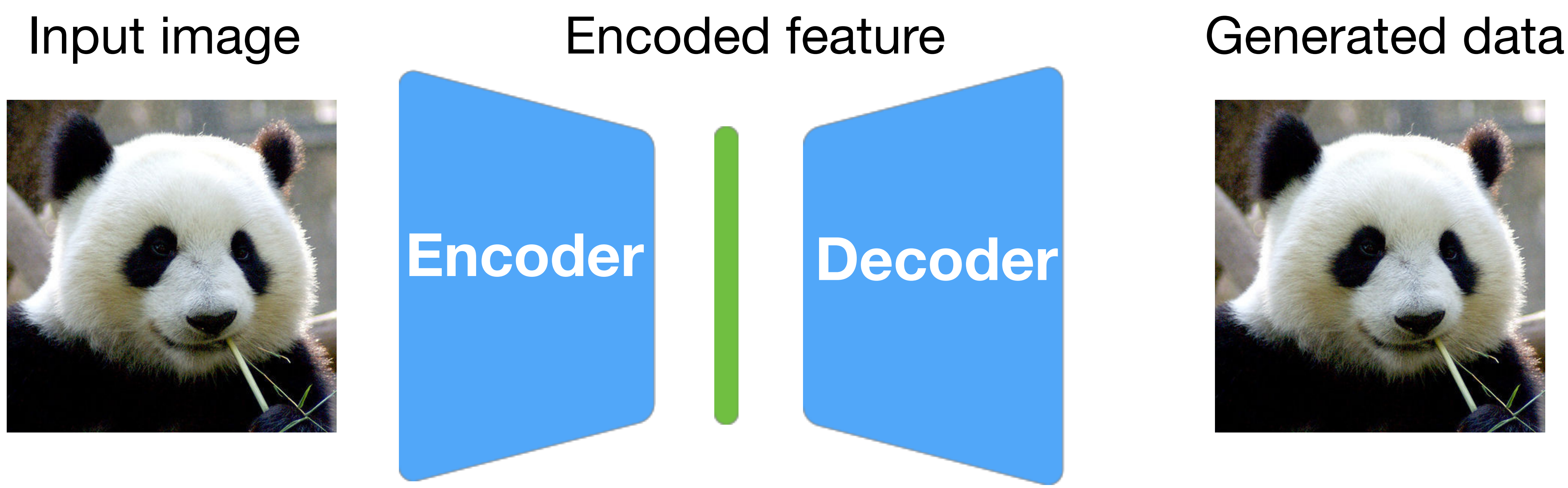
GAN Architecture



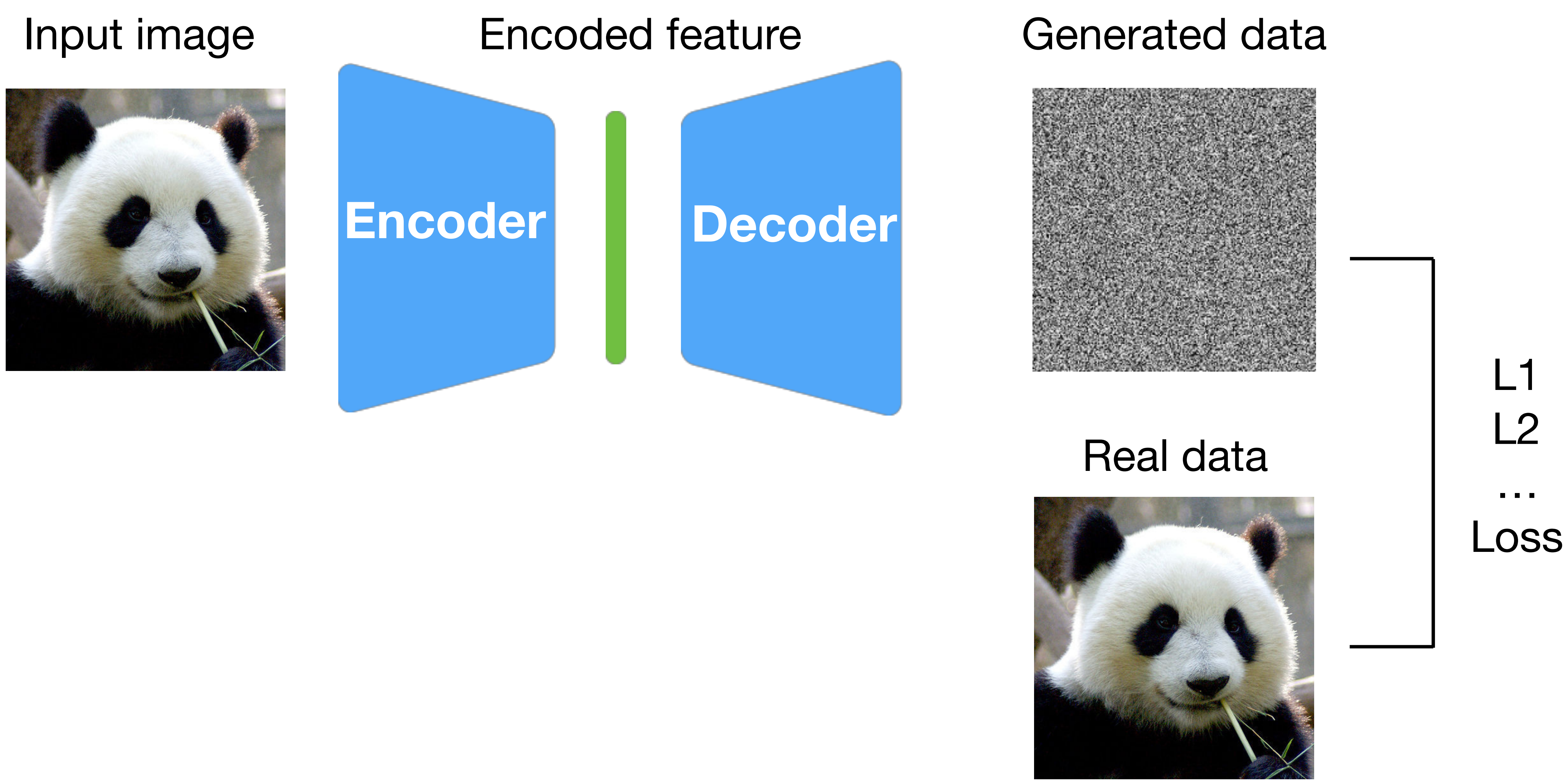
GAN with Autoencoder



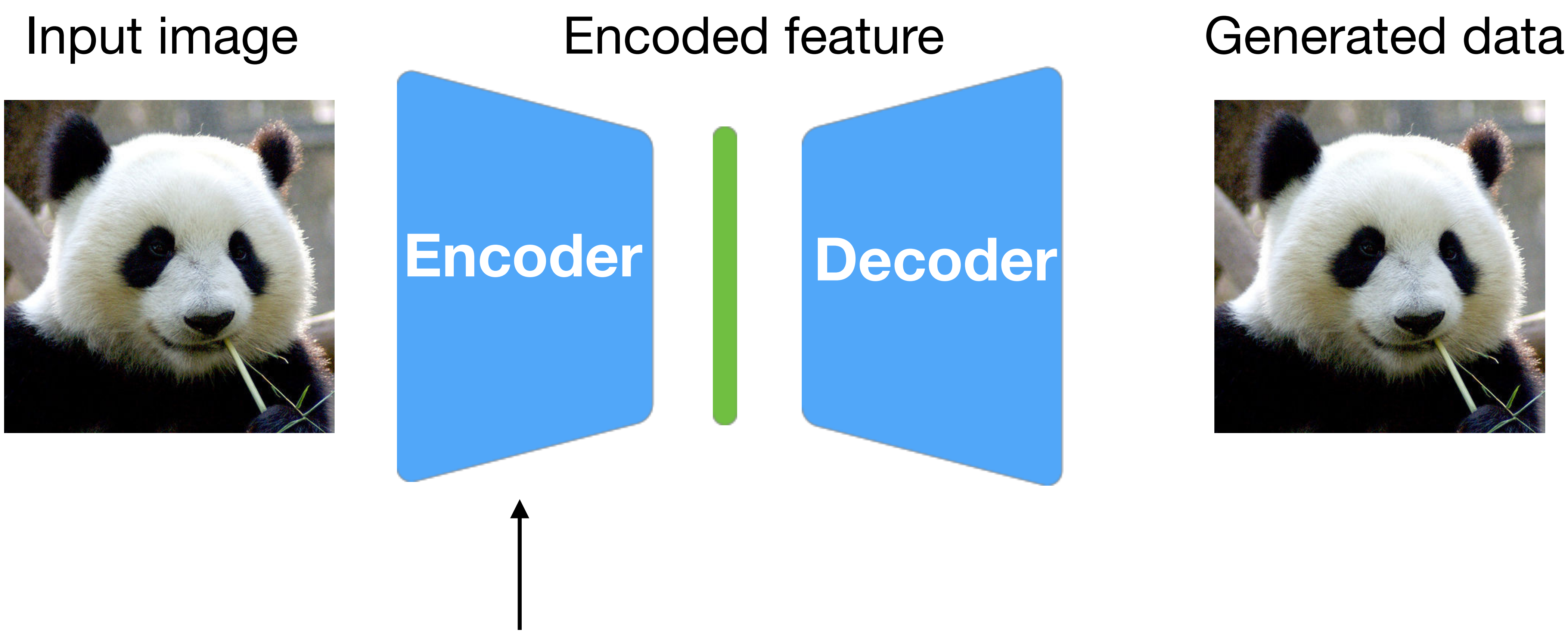
Autoencoder



Autoencoder

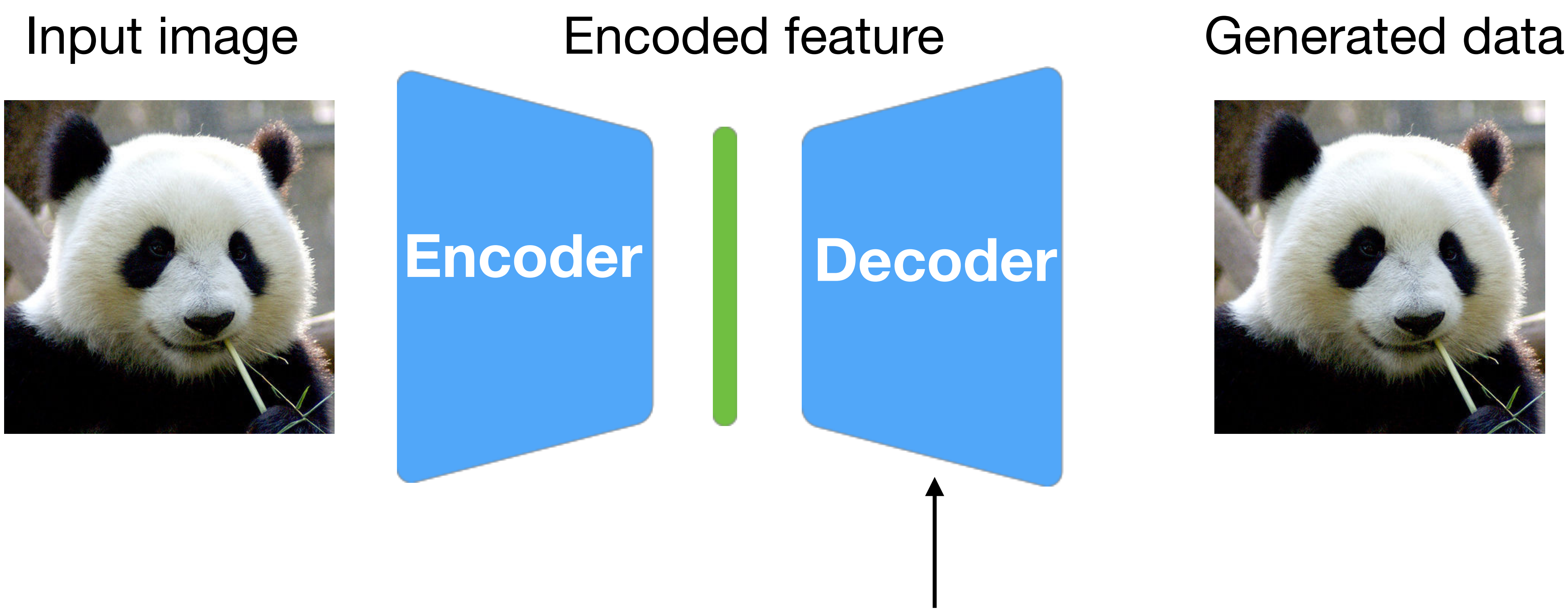


Autoencoder



- **Convolutional layer (tf.nn.conv2d). Set stride>1 to downsample**
- **Convolutional layer + pooling**
- **fully connected layer (tf.layers.dense)**

Autoencoder



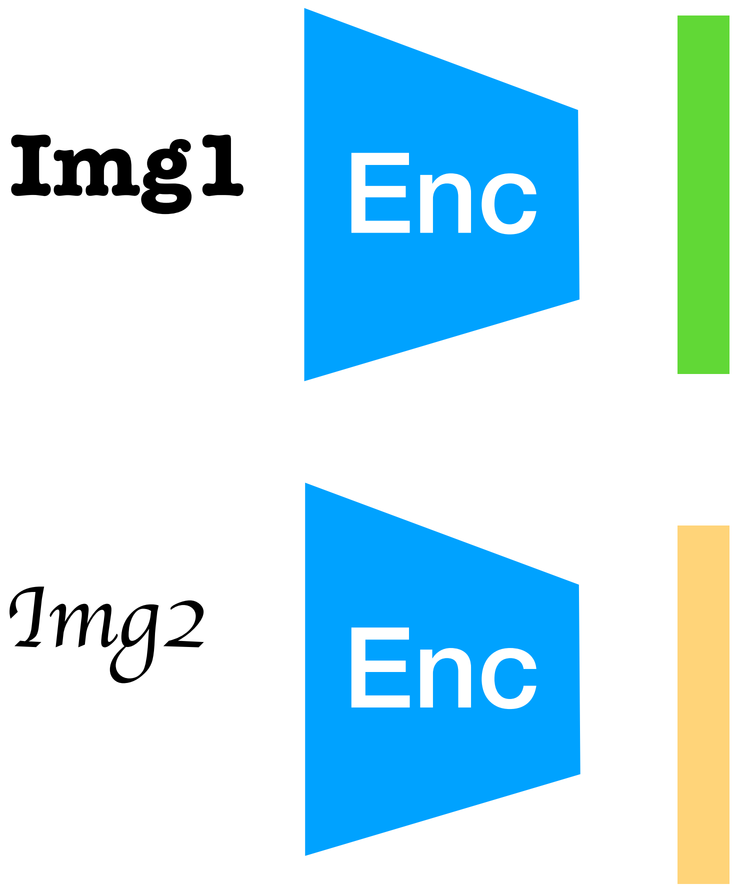
- deconvolutional layer (tf.nn.conv2d_transpose). Set stride>1 to upsample

- fully connected layer (tf.layers.dense) and reshape the output of last layer to the scale of the image

Disentangling

- Approach: Disentangling content and the rest (style, noise etc.) (supervise with weak label)

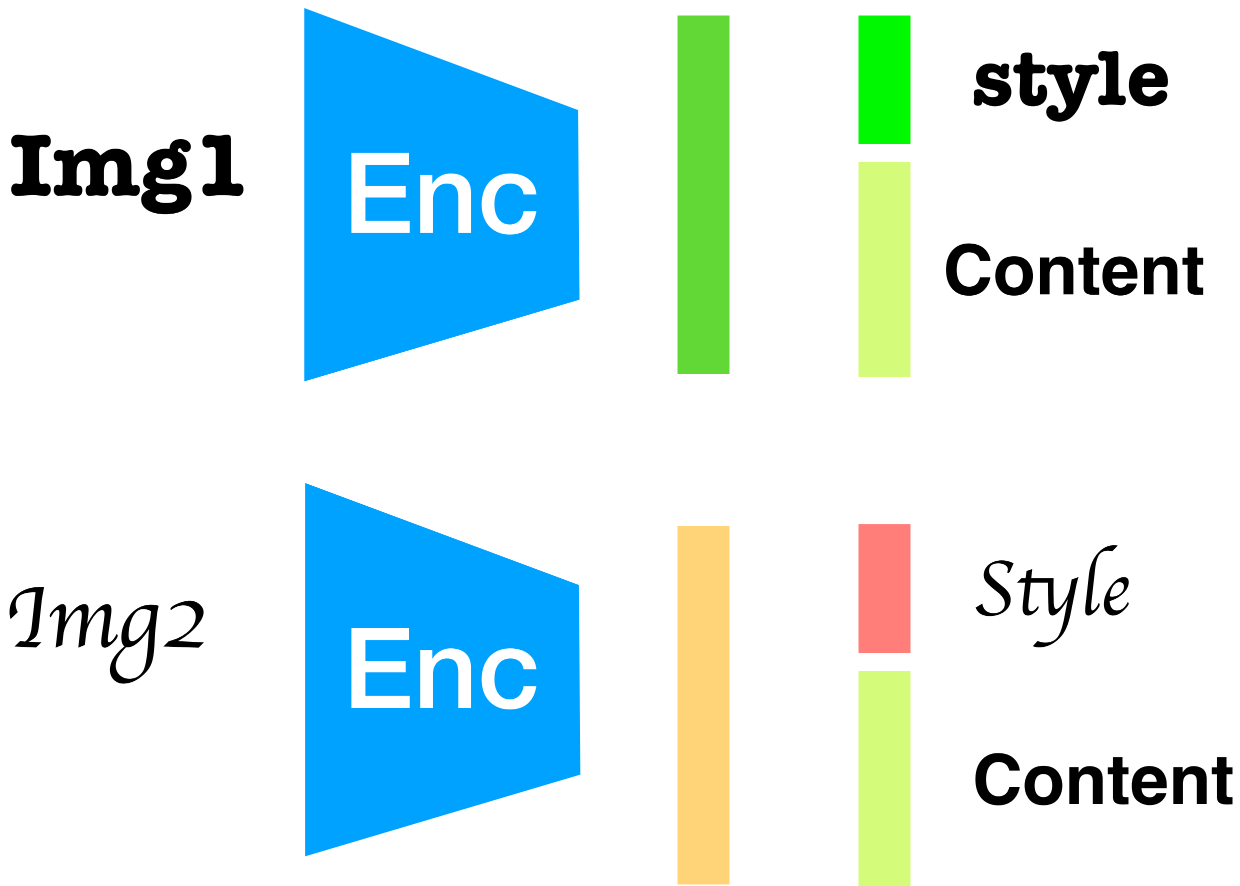
Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

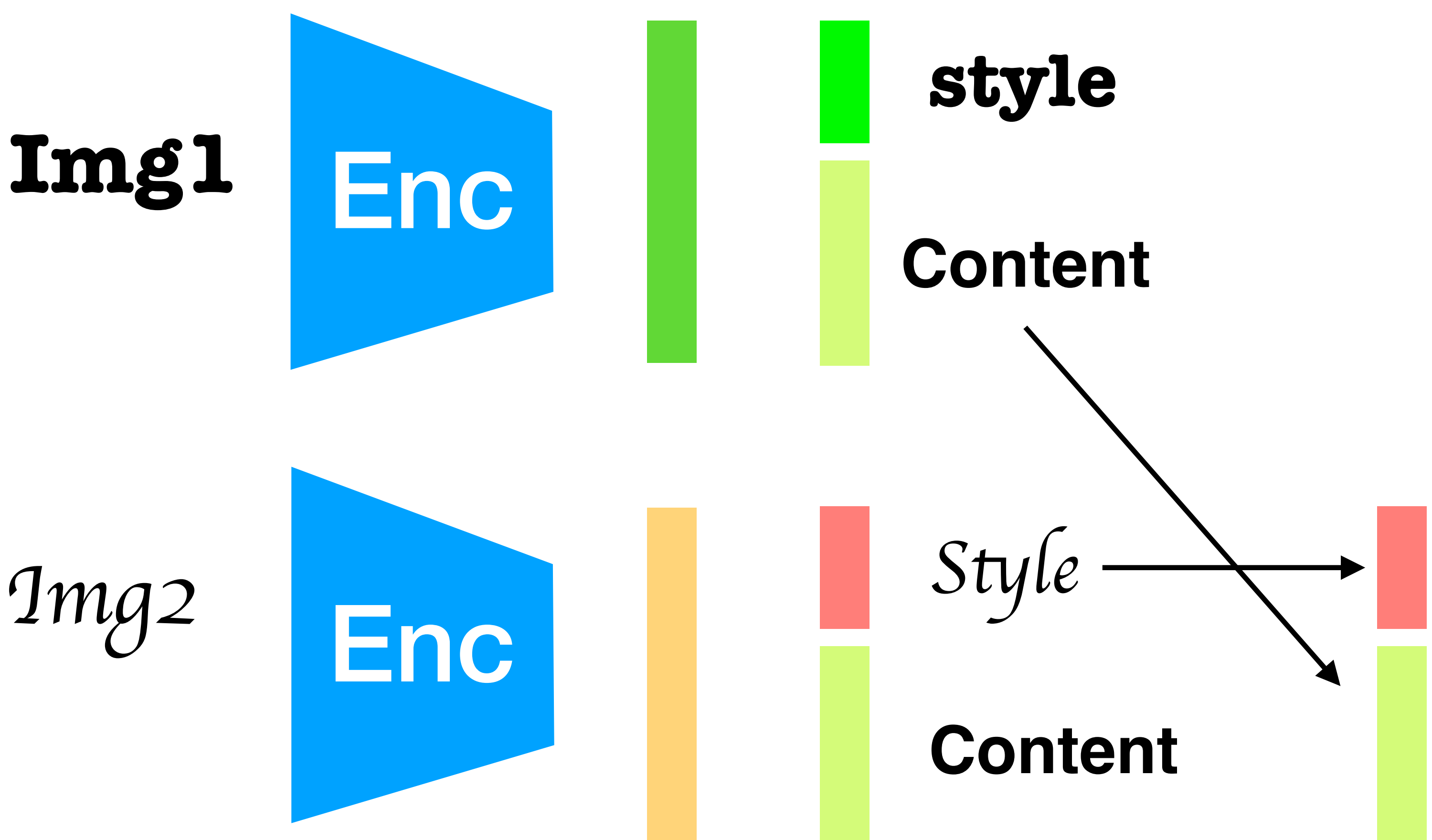
Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

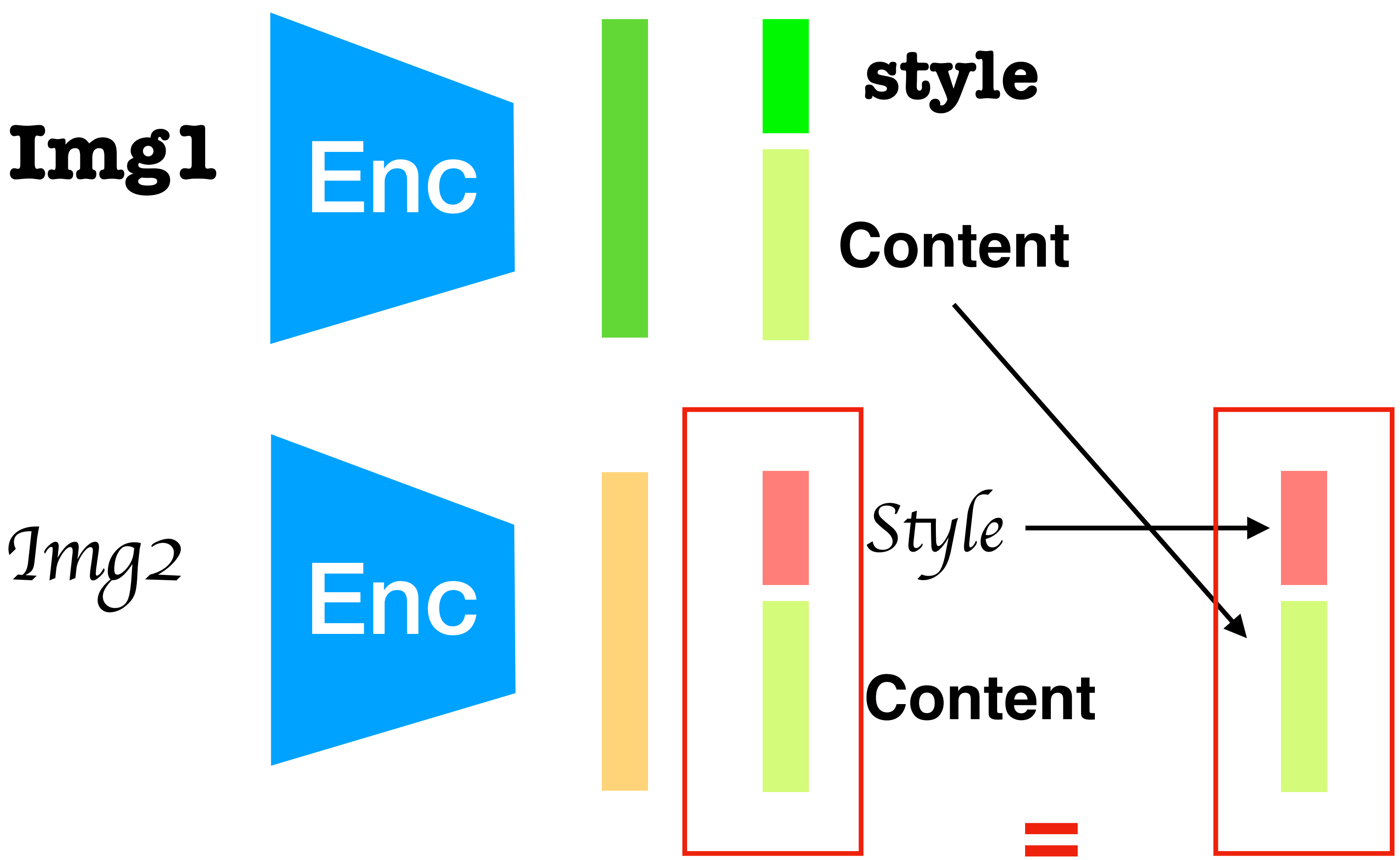
Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

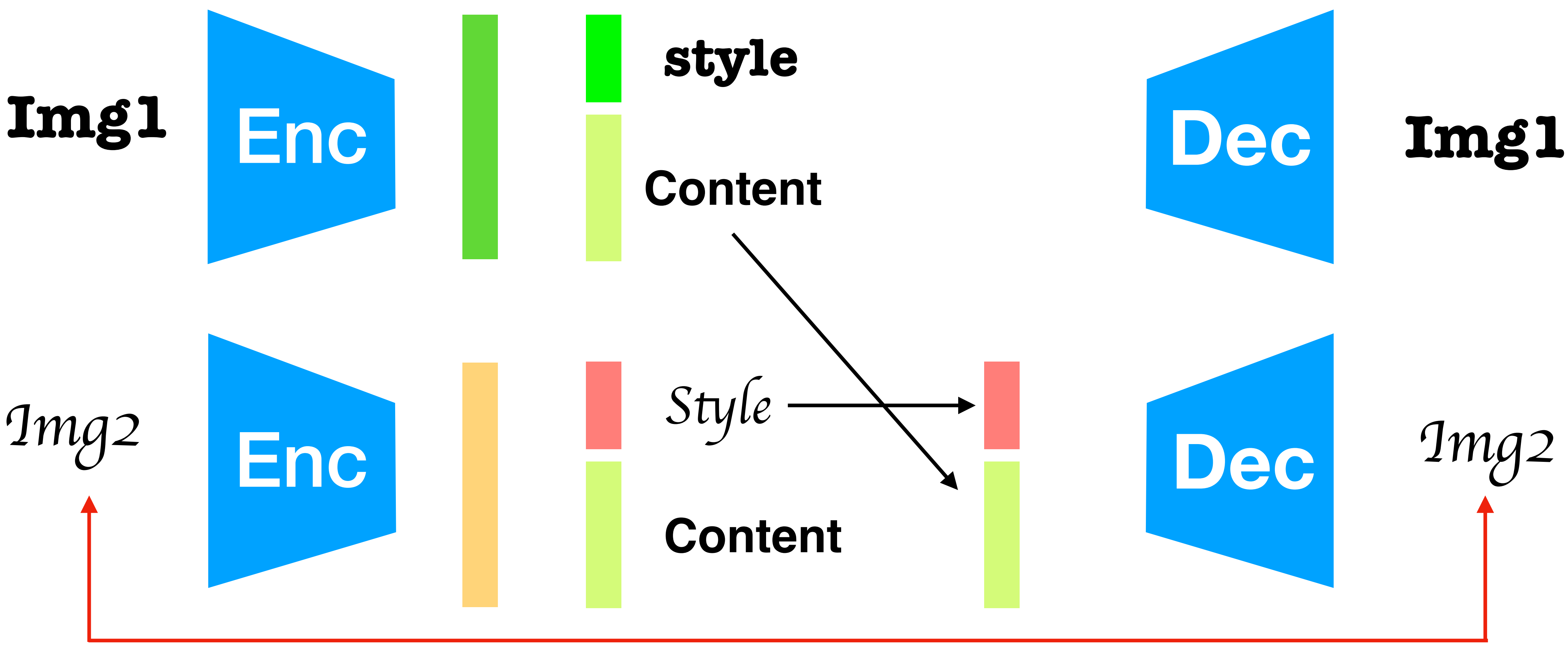
Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

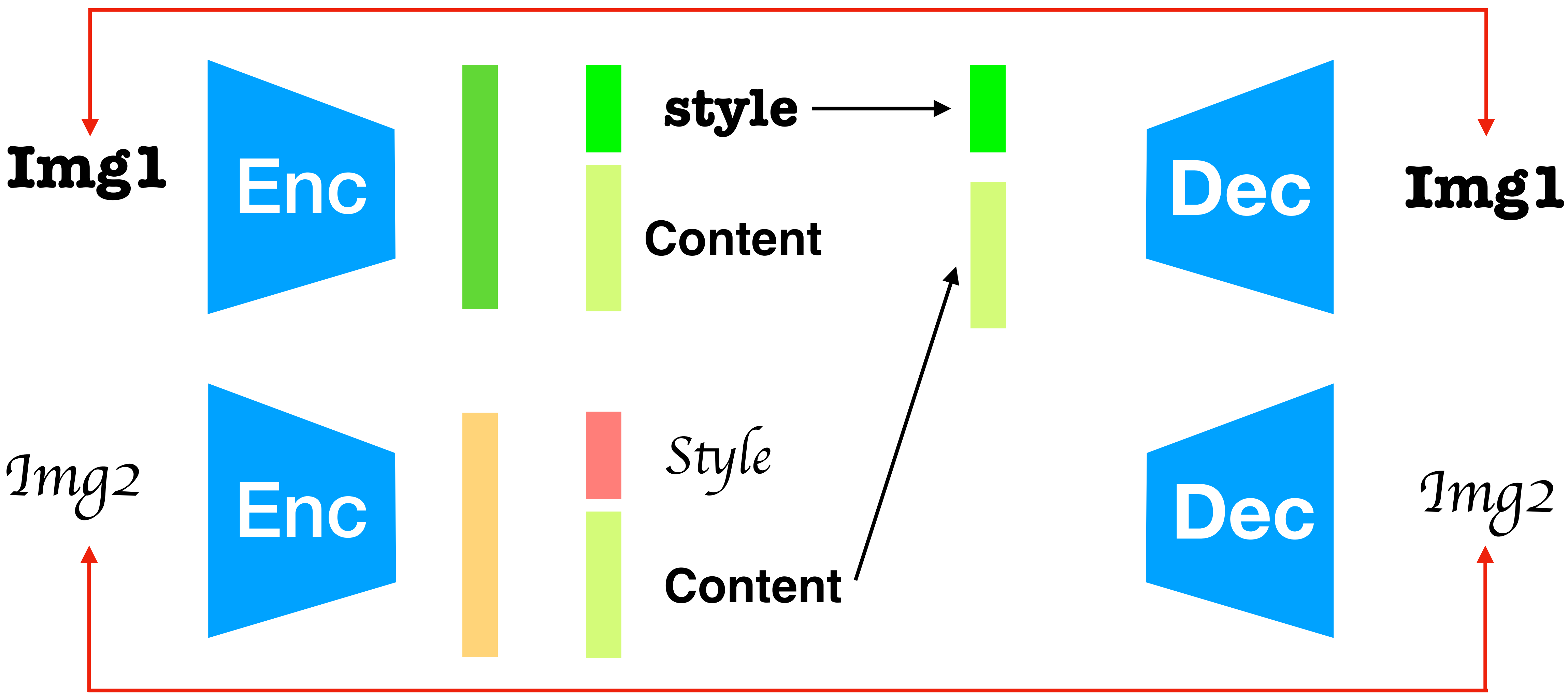
Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

Input: pair of text image, same content different style



Disentangling

- Approach: Disentangling content and the rest (style, noise etc.)

Output: Real data content + synthetic style