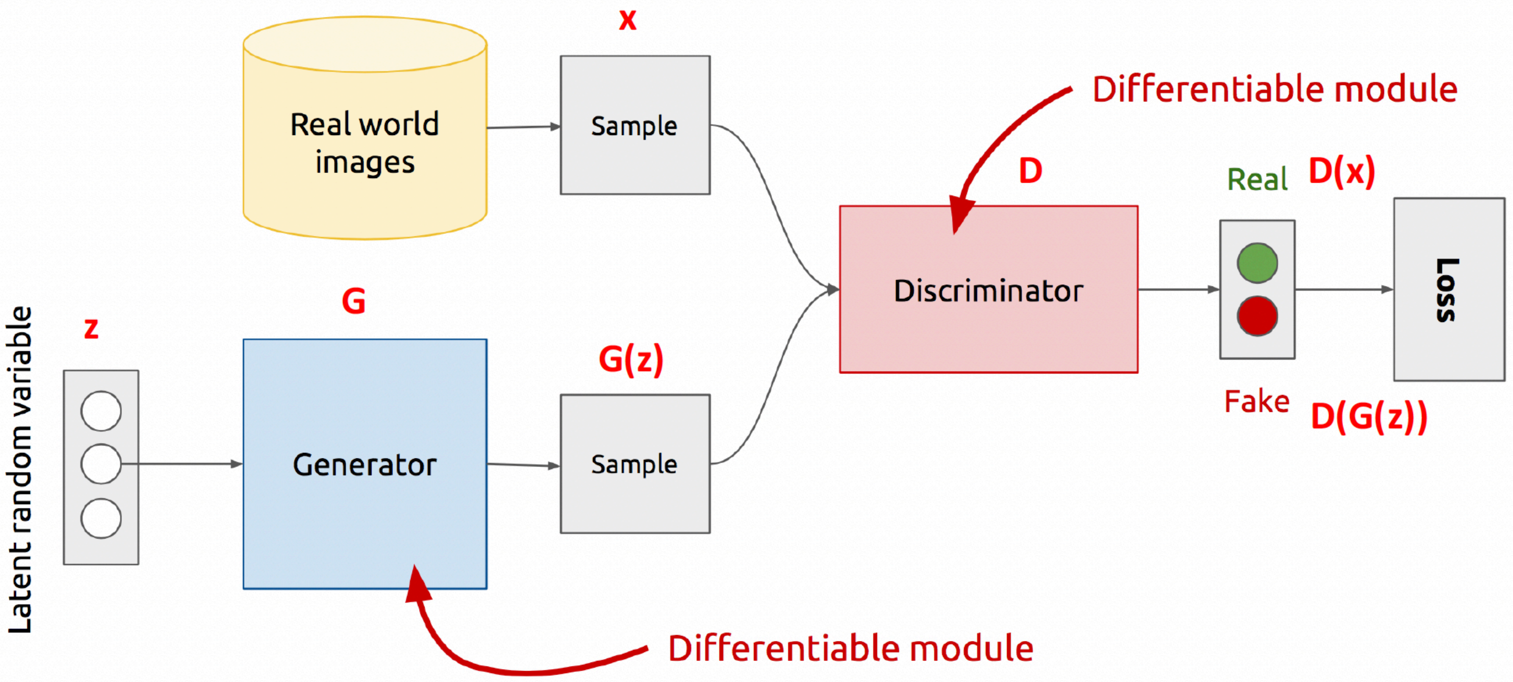
Generative-Adversarial Networks (GANs)

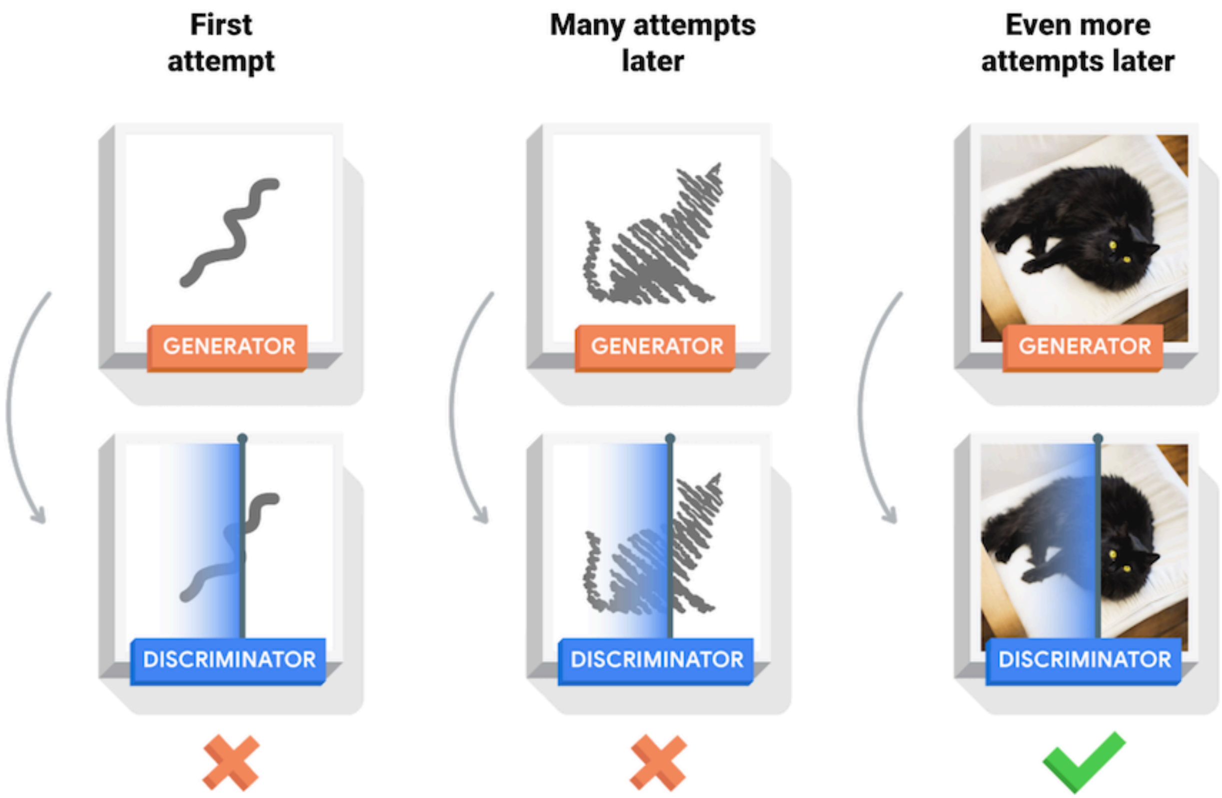
- **Generative**
  - Learn a generative model
- **Adversarial**
  - Trained in an adversarial setting
- **Networks**
  - Use Deep Neural Networks

Architecture

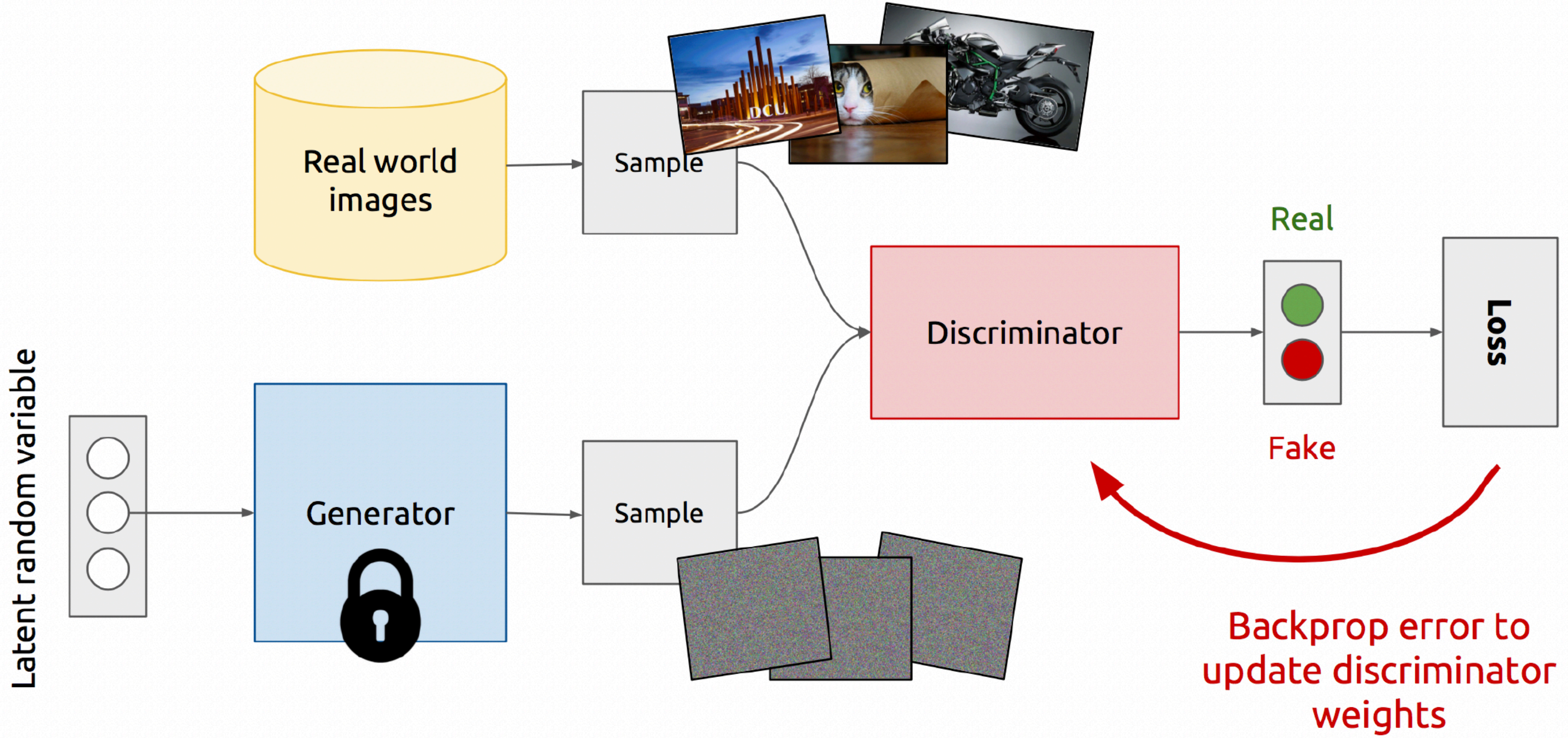


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

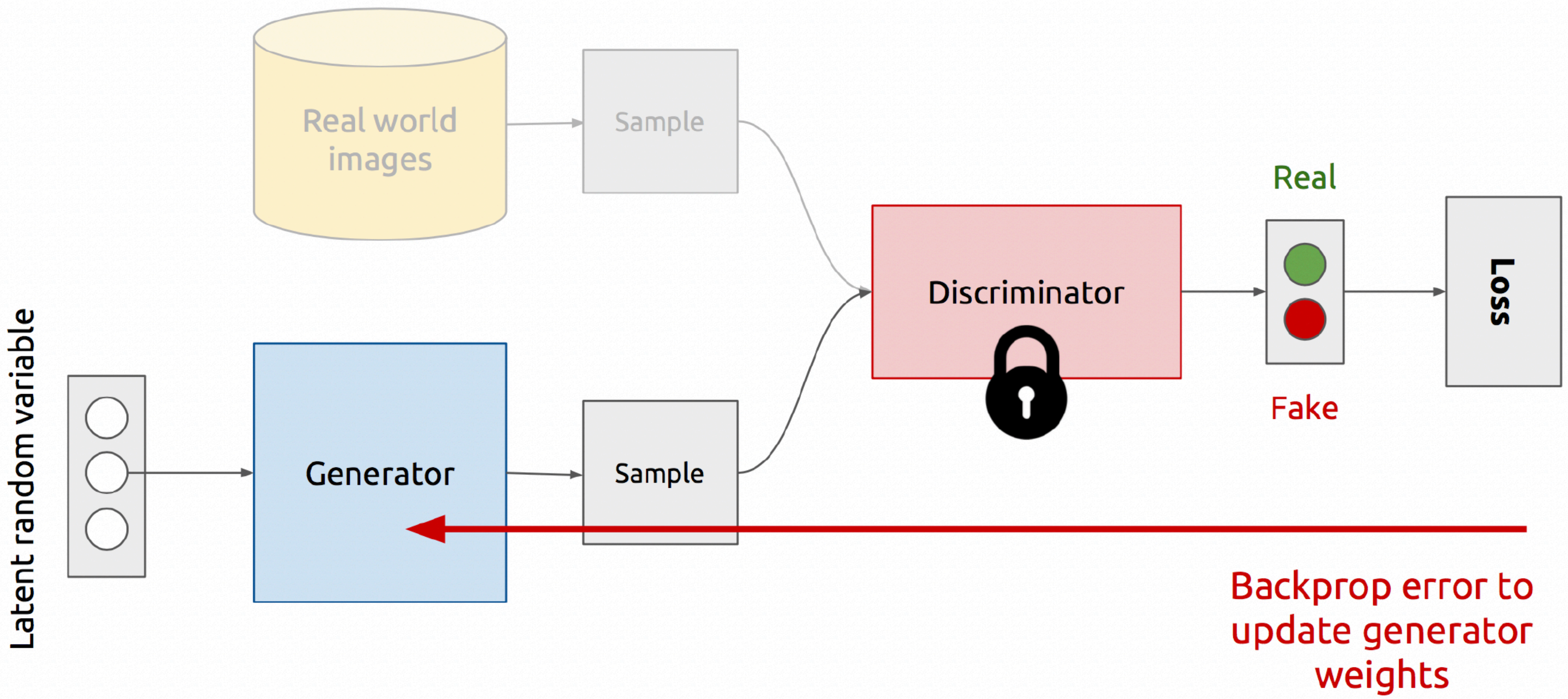
Basic principle



Training Discriminator



Training Generator



Example (simple GAN)

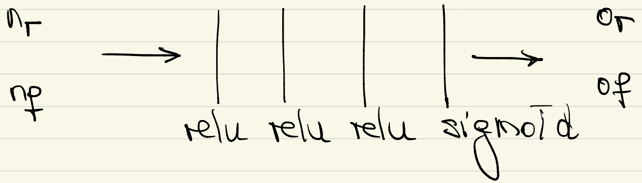
'Real' data from normal distribution $\mu = 5, \sigma^2 = 1$

Generator : 1-layer NN

Input : random number $U[-8,8]$

weights : $\underline{w}_G$

Discriminator , 4-layer NN



Loss function :

$$L_D = \frac{1}{2}\left(-\ln o_r - \ln(1-o_f)\right)$$

weights : $\underline{w}_D$

Possible iteration

w_G random

I Generator Real

0.01 example 3.8

Discriminator w_D random

$o_f \sim 0$ $o_r \sim 0$

$$L_d \sim -\ln o_r$$

w_D updated to increase o_r

II Generator loss function

$$L_G = -\ln o_f$$

adjust w_G to get larger o_f

Formulation

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \boxed{\mathbb{E}_{x\sim p(x)}[\log D(x)]} + \boxed{\mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]}$$

Algorithm

Discriminator updates

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

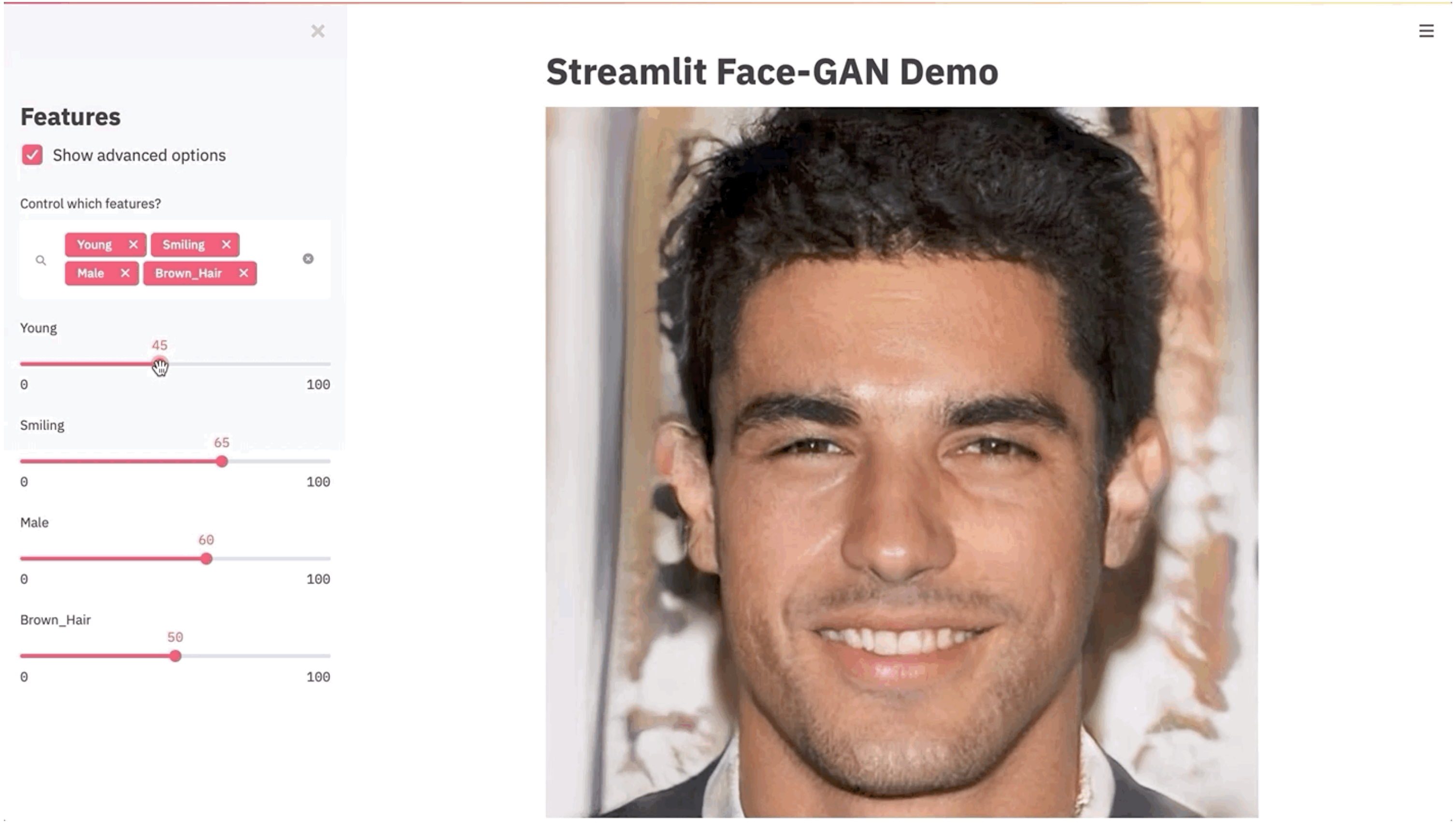
Generator updates

- 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:

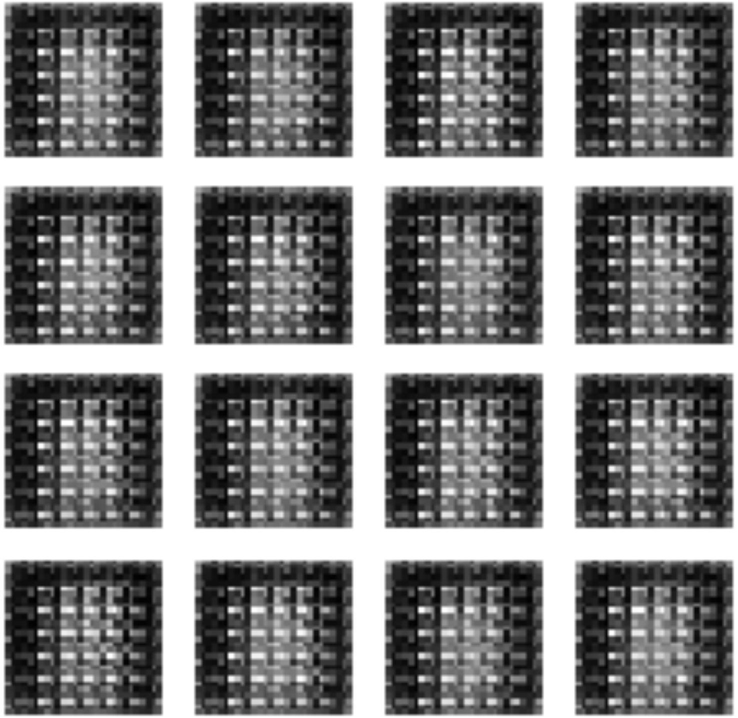
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

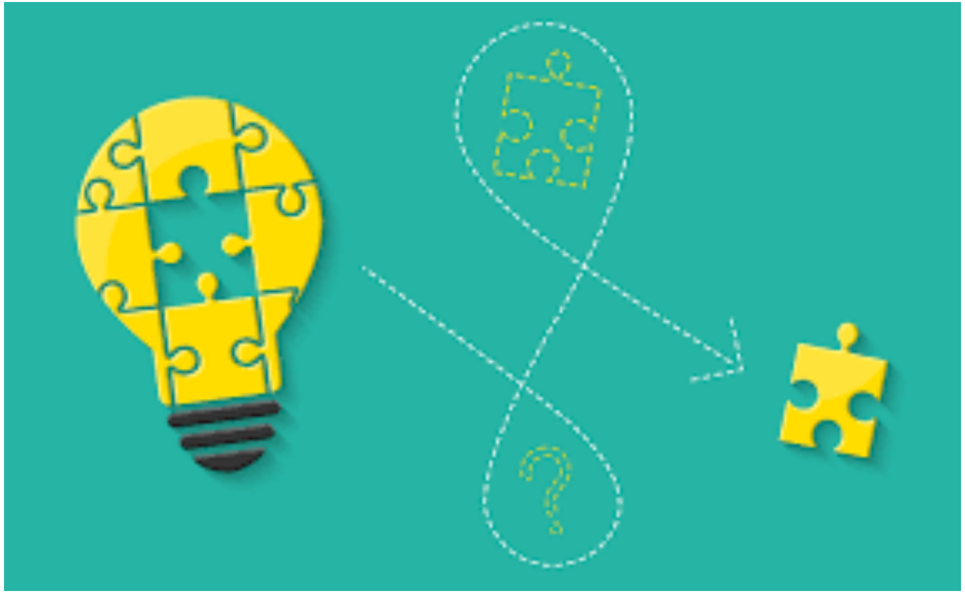
Example


Features
Show advanced options
Control which features?
Young
Smiling
Male
Brown_Hair
Young
45
0
100
Smiling
65
0
100
Male
60
0
100
Brown_Hair
50
0
100
Streamlit Face-GAN Demo

Example: MNIST



Exercise B7



LINK

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb