Generative Adversarial Networks

Why?

➢ 위키피디아에 GAN을 검색해보면?

- 2014년 6월에 Ian Goodfellow와 그의 동료들이 고안한 기계학습 프레임워크

Generative adversarial network

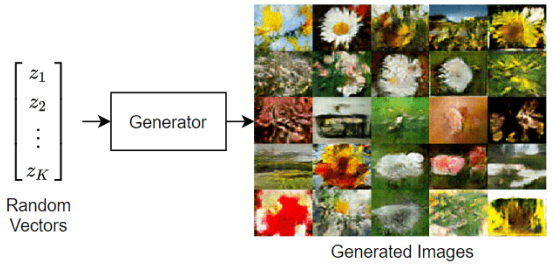
From Wikipedia, the free encyclopedia

Not to be confused with Adversarial machine learning.

A **generative adversarial network** (**GAN**) is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in June 2014.[1] Two neural networks contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.

What?

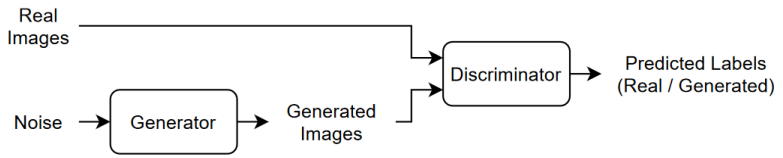
- Generative Adversarial Nets(Paper)
  - 생성가능한 적대적 신경망
  - Generator(생성자)와 Discriminator(판별자)로 구성
- Generator(생성자)
  - 입력: 노이즈
  - 출력: 가짜 데이터
- Discriminator(판별자)
  - 입력: 가짜 데이터 또는 진짜 데이터
  - 출력: 진위여부



[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

What?

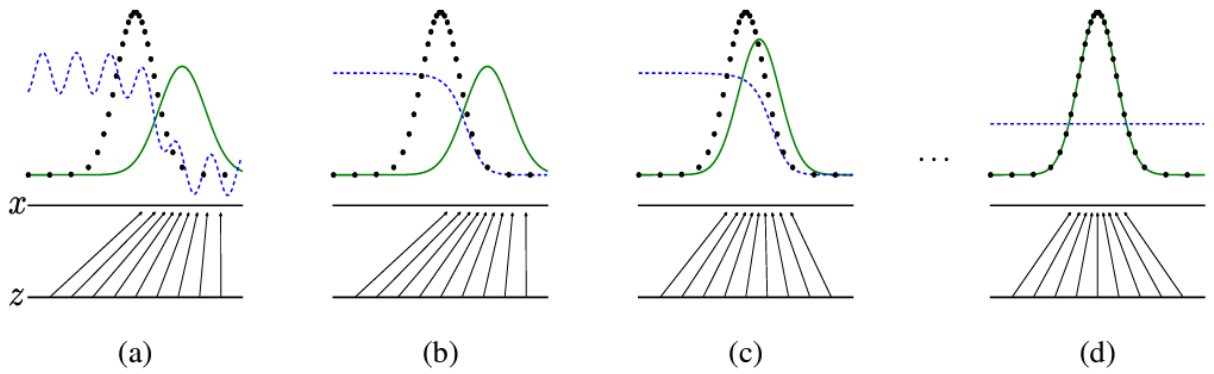
- Generative Adversarial Nets(Paper)
  - 생성가능한 적대적 신경망
  - Generator(생성자)와 Discriminator(판별자)로 구성
- Generator(생성자)
  - 입력: 노이즈
  - 출력: 가짜 데이터
- Discriminator(판별자)
  - 입력: 가짜 데이터 또는 진짜 데이터
  - 출력: 진위여부



[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

What?

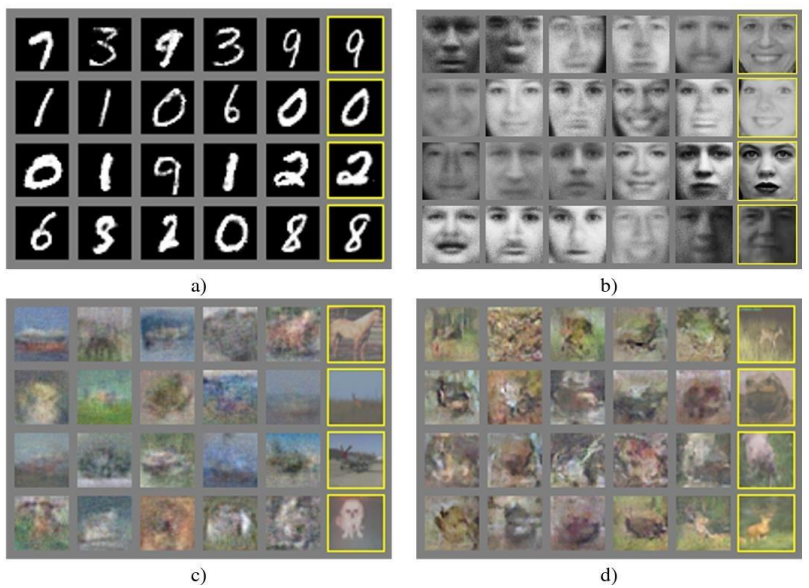
- 검정색 : 실제 데이터의 분포
- 초록색: 생성자가 생성한 데이터의 분포
- 파란색: 판별자가 판단하는 값



What?

➢ GAN이 생성한 이미지

- a : MNIST
- b: TFT(Toronto Face Database)
- c, d: CIFAR10



How?

- D and G play the following two-player minimax game with value function V (G,D)

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

How?

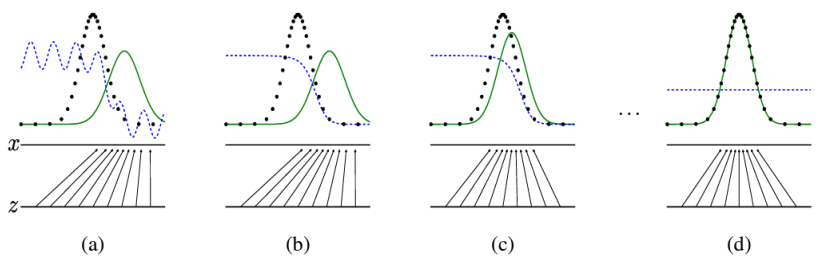
➢ D and G play the following two-player minimax game with value function V (G,D)

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

실제 데이터에 대한 확률분포에서 샘플링한 데이터

임의의 노이즈*에서 샘플링한 데이터

* : 일반적으로 가우시안 노이즈 사용



How?

- D and G play the following two-player minimax game with value function V (G,D)

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

실제 데이터가 진 짜라고 판단할 경우 1, 가 짜라고 판단할 경우 0을 출력

가짜 데이터가 진 짜라고 판단할 경우 1, 가 짜라고 판단할 경우 0을 출력

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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

max → $D(x) = 1$ → 실제 데이터를 진짜라고 판단

max → $D(G(z)) = 0$ → 가짜 데이터를 가짜라고 판단

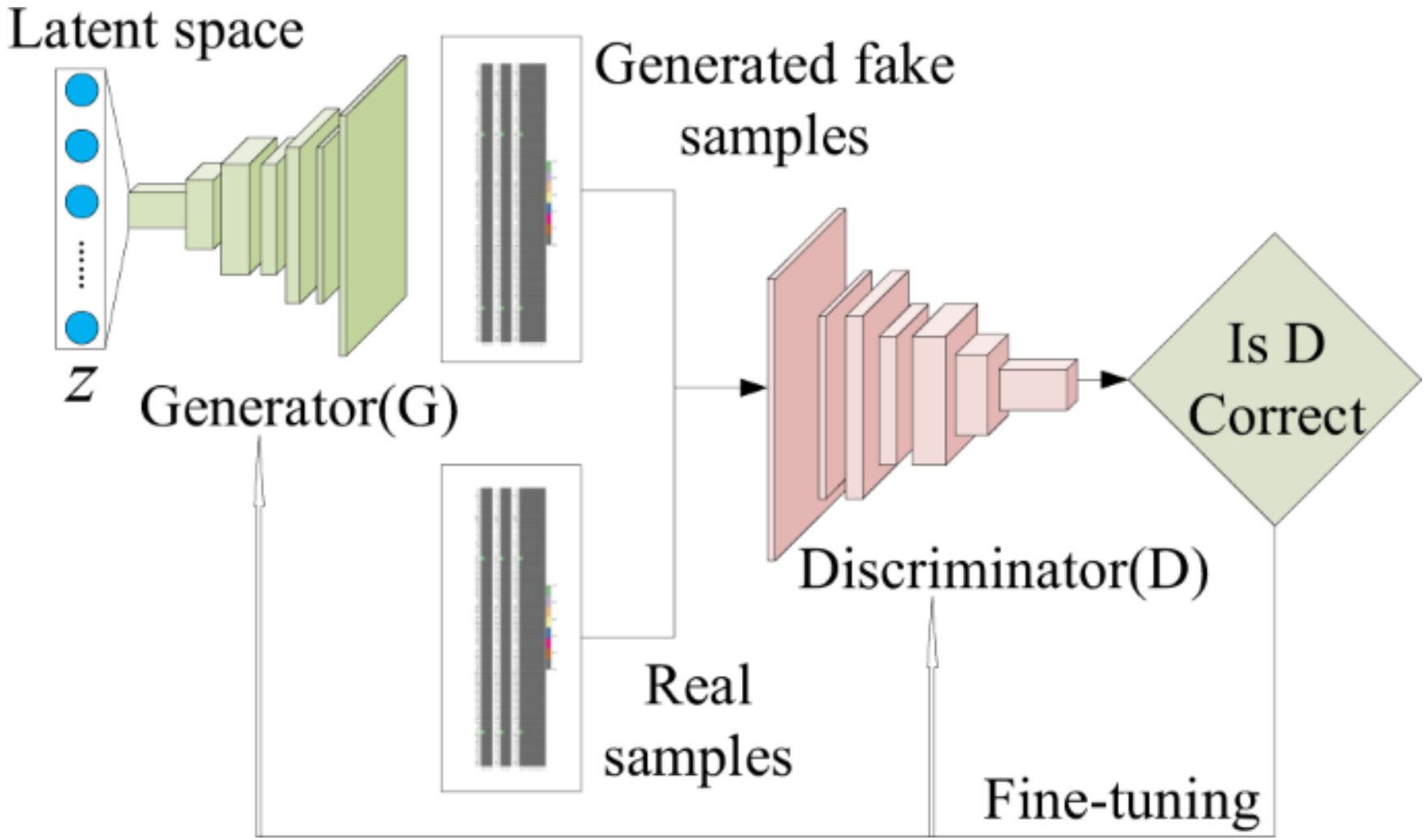
$$\underset{G}{minMax}V(D,G) = E_{x\text{-}p_{data}(x)}[logD(x)] + E_{z\text{-}p_z(z)}[log(1 - D(G(z))]$$

G D

x: 실제 데이터
z: 노이즈

$$minMaxV(\text{판별}, \text{생성}) = [log\,\text{판별}(x)] + [log(1 - \text{판별}(\text{생성}(z)))]$$

생성 판별


Latent space
z
Generator(G)
Generated fake samples
Real samples
Discriminator(D)
Is D Correct
Fine-tuning

How?

- D and G play the following two-player minimax game with value function V (G,D)

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

How?

- D and G play the following two-player minimax game with value function V (G,D)

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

How?

- D and G play the following two-player minimax game with value function V (G,D)

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

Conclusion

➢ GAN의 한계

- 학습시키는 것이 어렵다
  - 생성자와 판별자의 실력이 비슷한 경우에 두 모델이 균형있게 학습이 진행되는데, 그렇지 않은 경우 학습이 잘 진행되지 않음
- 사용된 생성자의 결과물 형태가 어떠한 과정을 통해 나왔는지 알 수 없다
- 새롭게 만들어진 데이터가 얼마나 정확한지 객관적으로 판단하기 어렵다
  - 주관적인 판단 필요

DCGAN

- Paper: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, Alec Radford, Luke Metz, Soumith Chintala, ICLR 2016

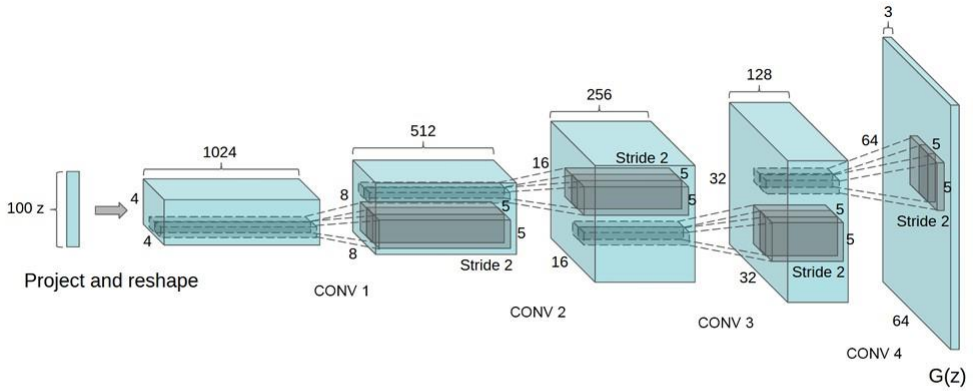
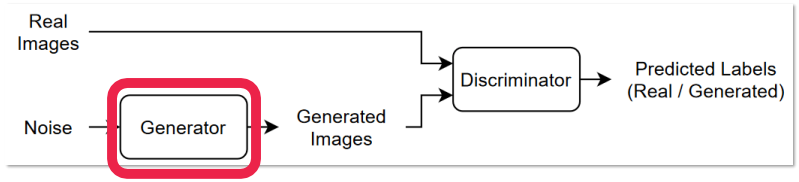
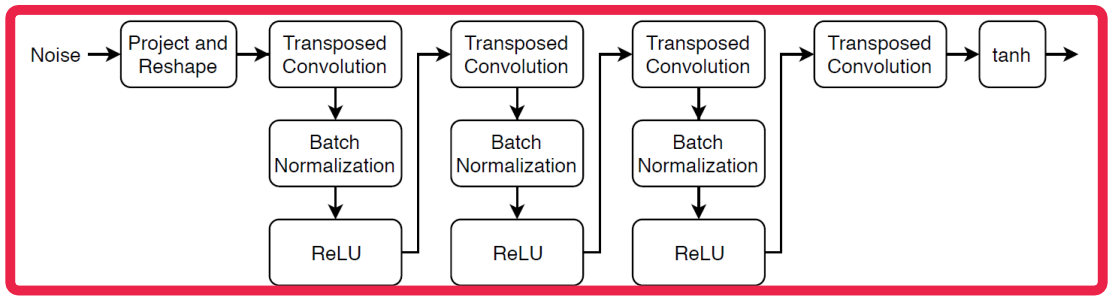


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN

➢ Architecture

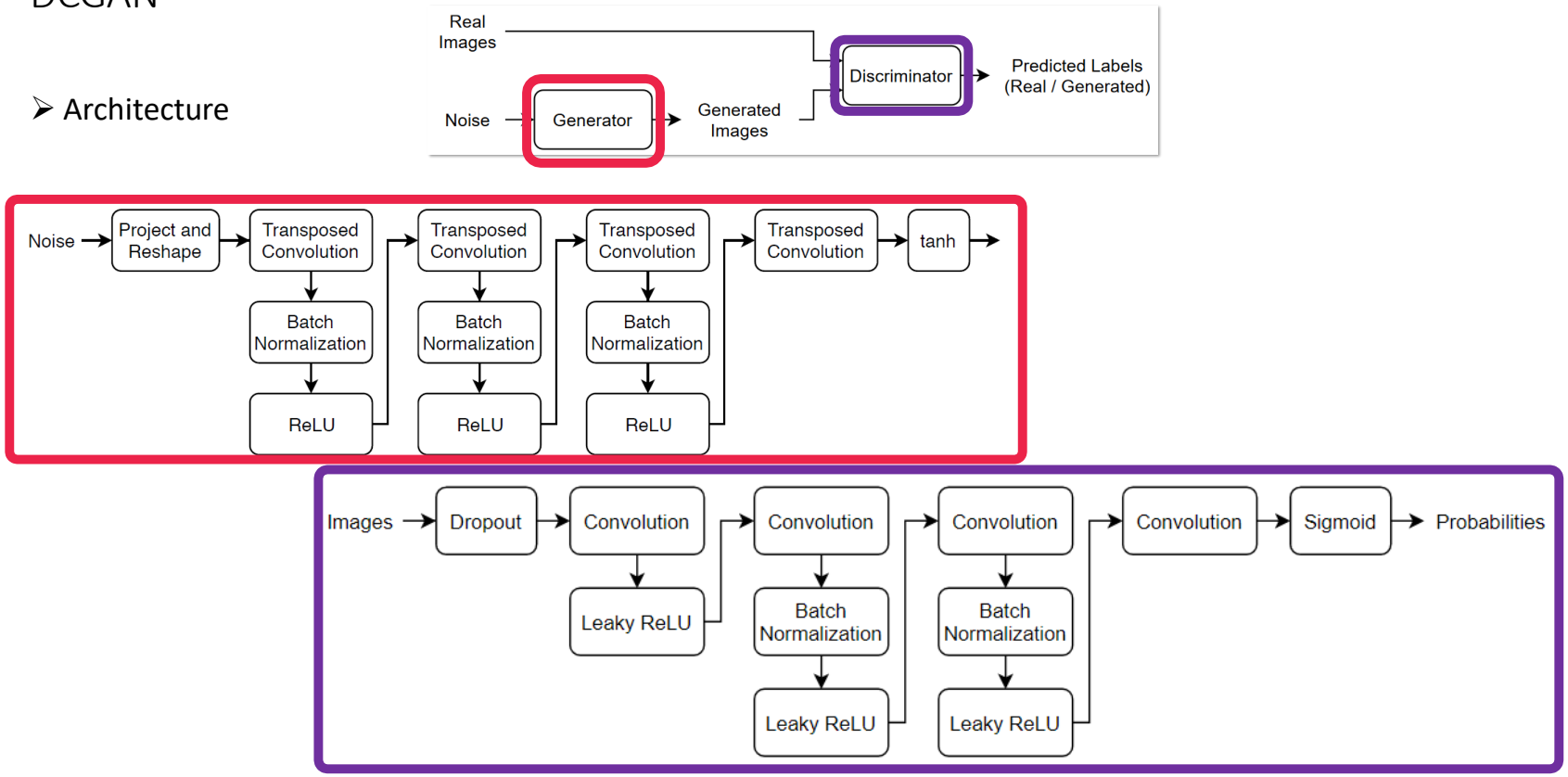




[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

DCGAN

➢ Architecture



[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html