Machine Learning

Contents

13.1. Discrimination and Generation

- Supervised learning
  - Training samples

$$D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$$

  - Objective

$$f : \boldsymbol{X} \times \Theta_1 \rightarrow \boldsymbol{Y}$$

- Unsupervised learning
  - Training samples

$$D = \{\boldsymbol{x}_i\}_{i=1}^{N}$$

  - Objective

$$f : \boldsymbol{X} \times \Theta_2 \rightarrow \boldsymbol{Y}$$

- Classification

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})}$$

  - Discriminator $f(\boldsymbol{x}; \boldsymbol{\theta}_d)$ to perform $p(\boldsymbol{y}|\boldsymbol{x})$
  - Discriminant function (w/o p.d.f. information)

$$f : \boldsymbol{X} \times \Theta_d \rightarrow \boldsymbol{Y}$$

- Machine learning : training from examples
- Generator : generation of samples from $p(\boldsymbol{x},\boldsymbol{y})$ or $p(\boldsymbol{x})$

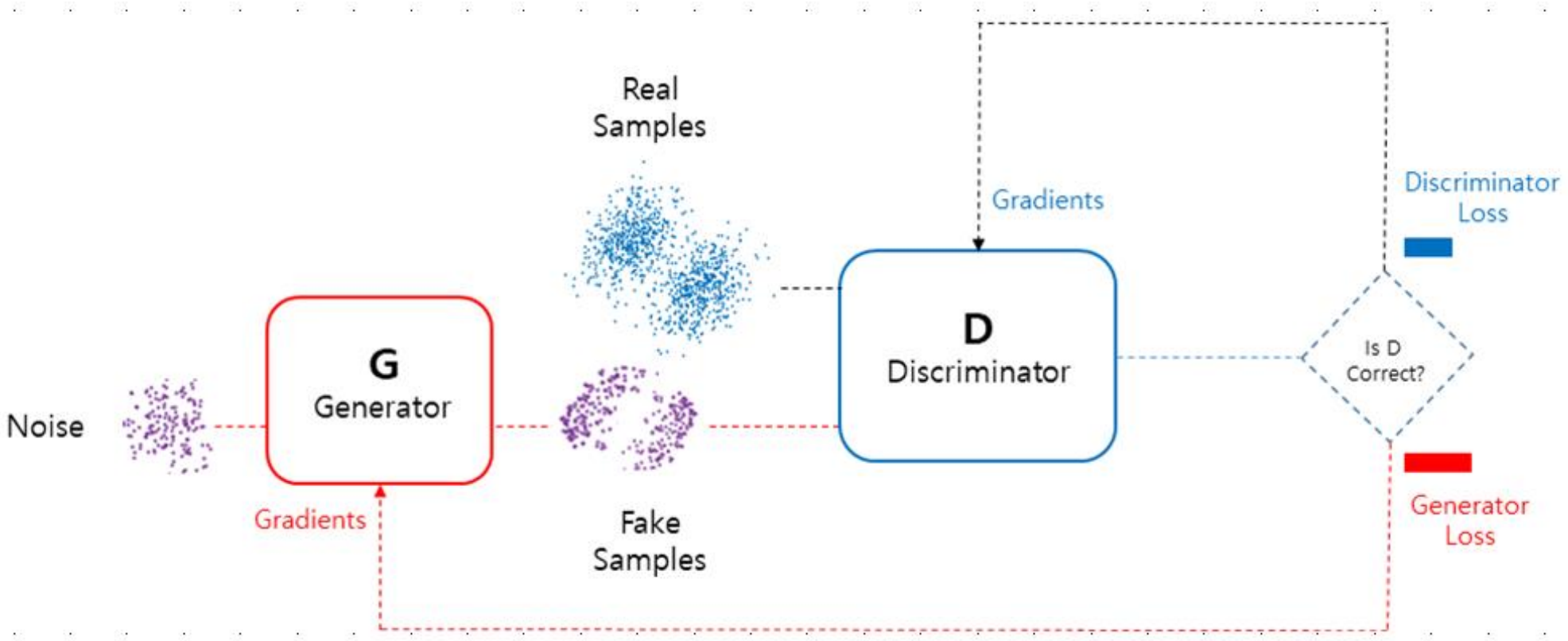
- Discriminator (supervised learning)

$$f : \boldsymbol{X} \times \Theta_d \rightarrow \boldsymbol{Y}$$
$$\min_{\theta_d} L(\boldsymbol{y},\boldsymbol{y}') = \min_{\theta_d} L(\boldsymbol{y}, f(\boldsymbol{x};\boldsymbol{\theta}_d))$$

- Generator (unsupervised learning)

$$f : \boldsymbol{X} \times \Theta_g \rightarrow \boldsymbol{X}$$
$$\min_{\theta_g} L(\boldsymbol{x},\boldsymbol{x}') = \min_{\theta_g} L(\boldsymbol{x}, f(\boldsymbol{x};\boldsymbol{\theta}_g))$$

13.2. GAN(Generative Adversarial Network)



- Generator
  - It aims to generate new data similar to the expected one. The Generator could be asimilated to a human art forger, which creates fake works of art.

- Discriminator
  - This model's goal is to recognize if an input data is '*real*' — belongs to the original dataset — or if it is '*fake*' — generated by a forger. In this scenario, a Discriminator is analogous to the police (or an art expert), which tries to detect artworks as truthful or fraud.

- **How do these models interact?**
  - It can be thought of the Generator as having an adversary, the Discriminator. The Generator (forger) needs to learn how to create data in such a way that the Discriminator isn't able to distinguish it as fake anymore. The competition between these two teams is what improves their knowledge, until the Generator succeeds in creating realistic data.

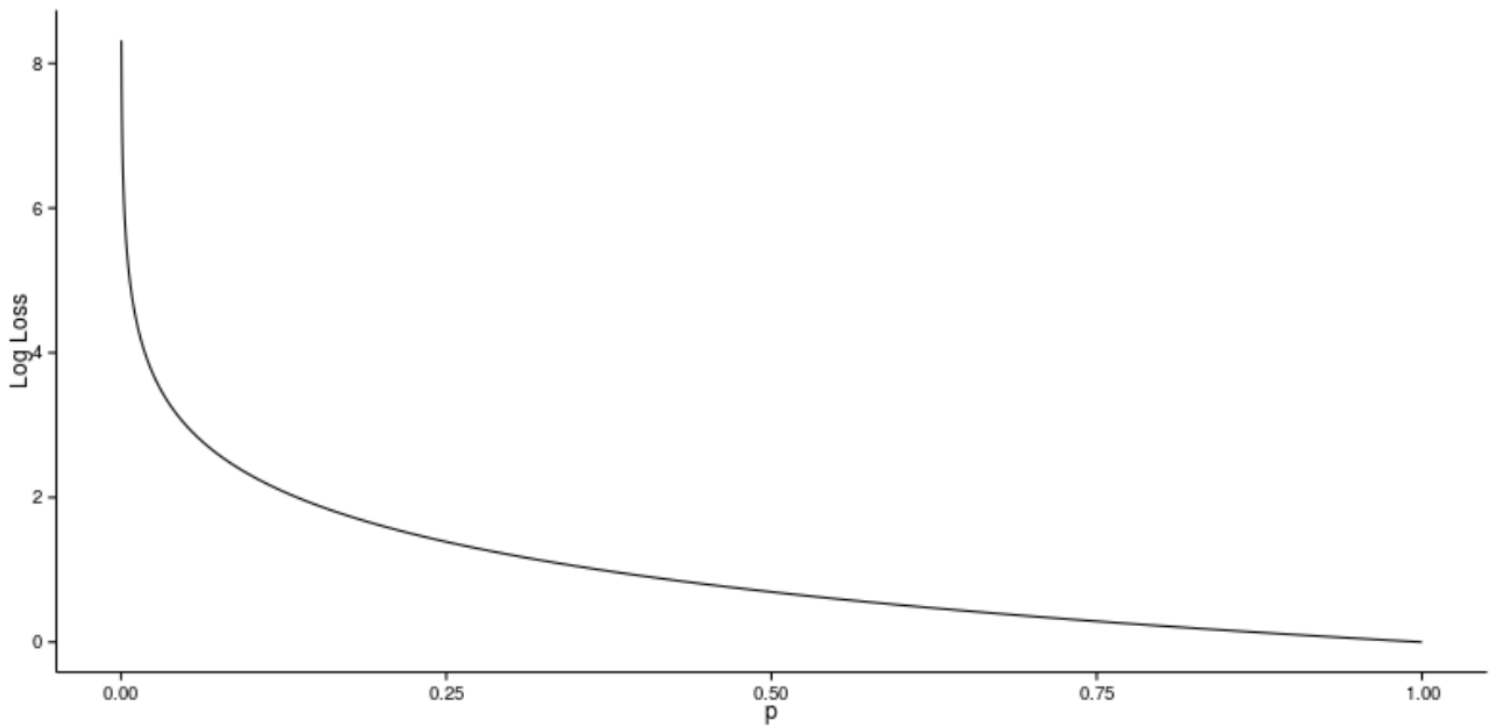
13.3. Mathematical Modelling of GAN

- Feed-forward neural network: Universal approximator
- Discriminator
  - It's weights are updated as to **maximize** the probability that any real data input ***x*** is classified as belonging to the real dataset, while **minimizing** the probability that any fake image is classified as belonging to the real dataset. In more technical terms, the loss/error function used **maximizes the function *D(x)*, and it also minimizes *D(G(z))***.

- Generator
  - A neural network $G(\mathbf{z}, \boldsymbol{\theta_1})$ is used to model the Generator mentioned above. It's role is mapping input noise variables **z** to the desired data space ***x*** (say images). Conversely, a second neural network $D(\mathbf{x}, \boldsymbol{\theta_2})$ models the discriminator and outputs the **probability that the data came from the real dataset,** in the range (0,1). In both cases, $\boldsymbol{\theta_i}$ represents the weights or parameters that define each neural network.
  - The **Generator's** weight's are optimized to maximize the probability that any fake image is classified as belonging to the real datase. Formally this means that the loss/error function used for this network **maximizes *D(G(z))***.

$$\min_G \max_D E_{p_{data}(\boldsymbol{x})}[D(\boldsymbol{x})] + E_{p_z(z)}[1 - D(G(\boldsymbol{z}))] \tag{13.3.1}$$

$$\min_G \max_D V(D, G) = E_{p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{p_z(z)}[\log(1 - D(G(\boldsymbol{z})))] \tag{13.3.2}$$



Log Loss Visualization: Low probability values are highly penalized

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.

명제 1

생성기 G가 고정되면 최적의 판별기는

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{13.3.5}$$

이다.

증명

생성기 G가 주어지면 판별기 D는 가치함수 $V(D,G)$

$$\begin{aligned} V(D,G) &= \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x}) \log D(\boldsymbol{x}) d\boldsymbol{x} + \int_{z} p_z(z) \log(1 - D(G(z))) dz \\ &= \int_{\boldsymbol{x}} [p_{data}(\boldsymbol{x}) \log D(\boldsymbol{x}) d\boldsymbol{x} + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))] d\boldsymbol{x} \end{aligned} \tag{13.3.6}$$

를 최대화시키도록 학습된다. $a,b \in R^2/\{0,0\}$에 대하여 $y \rightarrow a\log y + b\log(1-y)$는 $[0,1]$ 구간에서 $\frac{a}{a+b}$에서 최댓값을 가진다. 따라서, 생성기 G가 고정되면 $V(D,G)$의 최대점은 식 (13.3.5)가 된다□

$$
\begin{aligned}
C(G) &= \max_D V(D,G) \\
&= E_{p_{data}(\boldsymbol{x})}[\log D_G^*(\boldsymbol{x})] + E_{p_z(z)}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= E_{p_{data}(\boldsymbol{x})}[\log D_G^*(\boldsymbol{x})] + E_{p_g(\boldsymbol{x})}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= E_{p_{data}(\boldsymbol{x})}\left[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + E_{p_g(\boldsymbol{x})}\left[\log \frac{p_g(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
\tag{13.3.7}
$$

정리 1

학습기준 $C(G)$의 최저치는 $p_g(\boldsymbol{x}) = p_{data}(\boldsymbol{x})$일 때에만 얻어지며 그 값은 $-\log(4)$이다.

증명

$p_g(\boldsymbol{x}) = p_{data}(\boldsymbol{x})$이면 식 (13.3.5)에서 $D_G^*(\boldsymbol{x}) = 1/2$이고, 식 (13.3.7)에서 $C(G) = -\log(4)$이다. 이를 염두에 두고 식 (13.3.7)을 다시 정리하면

$$\begin{aligned} C(G) &= -\log 4 + KL\left(p_{data}(\boldsymbol{x}) \,\|\, \frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}\right) + KL\left(p_g(\boldsymbol{x}) \,\|\, \frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}\right) \\ &= -\log 4 + 2JSD(p_{data}(\boldsymbol{x}) \,\|\, p_g(\boldsymbol{x})) \end{aligned} \tag{13.3.8}$$

이 된다. 여기서, KL은 쿨백-라이블러 발산(Kullback-Leibler Divergence)이고 JSD는 젠센-샤논 발산(Jensen-Shannon Divergence)이며

$$JSD(p\|q) = \frac{1}{2}KL(p\|r) + \frac{1}{2}KL(q\|r) \tag{13.3.9}$$

와 같이 정의된다. 여기서, $r = \frac{p+q}{2}$이다. 두 분포 사이의 JSD는 항상 양의 값을 가지며, 두 분포가 같을 때만 영의 값을 가진다. 따라서, 식 (13.3.8)의 최소치는 $p_g(\boldsymbol{x}) = p_{data}(\boldsymbol{x})$일 때 $C^*(G) = -\log(4)$로 얻어진다. 즉, 생성기가 완벽하게 데이터 분포를 재현한 것이다□

예제 13.3-1

정리 1을 증명하기 위해서는 다음과 같이 정의되는 쿨백-라이블러 발산(Kullback-Leibler Divergence)에 대한 이해가 필수적이다.

$$KL(p\|q) = \int p(x)\log\frac{p(x)}{q(x)}dx$$

이때, $KL(p\|q) \geq 0$이고 $p(x) = q(x)$일 때 최소임을 보여라.

풀이

Jensen의 부등식을 이용하면

$$-KL(p\|q) = \int p(x)\log\frac{q(x)}{p(x)}dx \leq \log\int p(x)\frac{q(x)}{p(x)}dx = \log\int q(x)dx = \log 1 = 0$$

이다. 즉, $KL(p\|q) \geq 0$ 이다. 그리고, 등식은 $p(x) = q(x)$일 때만 성립한다.

13.4. Coding a GAN

- Leaky ReLU

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \mathrm{negative-slope} \times x & \text{otherwise} \end{cases} \tag{13.4.1}$$

- Training Discriminator
  - Maximizing

$$\frac{1}{m}\sum_{i=1}^{m}\left[\log D(\boldsymbol{x}^{(i)}) + \log\left(1 - D(G(\boldsymbol{z}^{(i)}))\right)\right] \tag{13.4.2}$$

$$E_{CE} = -\left[t_i \log y_i + (1 - t_i)\log\left(1 - y_i\right)\right] \tag{13.4.3}$$

- Training a Generator
  - Minimizing

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D(G(\boldsymbol{z}^{(i)}))\right) \tag{13.4.4}$$

13.5. DCGAN

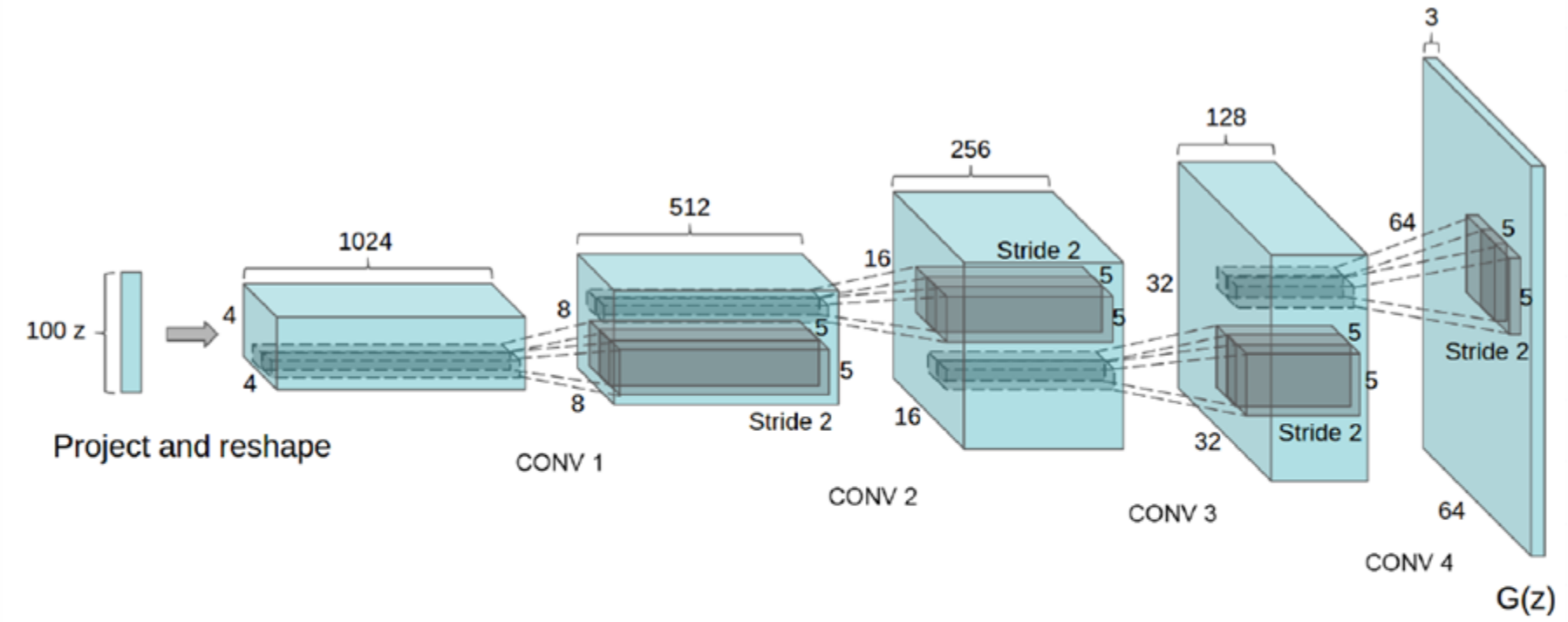


그림 13.2. 심층 컨볼루션 생성대립 회로망(DCGAN)의 생성기 모델. 100차원의 균일분포 잡음벡터 입력에 대하여 4층의 이동폭 2인 이항 컨볼루션 필터가 적용되어 64×64 영상을 생성함 [A. Radford et al. 2016].

그림 13.3. 심층 컨볼루션 생성대립 회로망(DCGAN)을 LSUN(Large-Scale Scene Understanding) 데이터에 적용하여 생성된 침실 영상 [A. Radford et al. 2016].

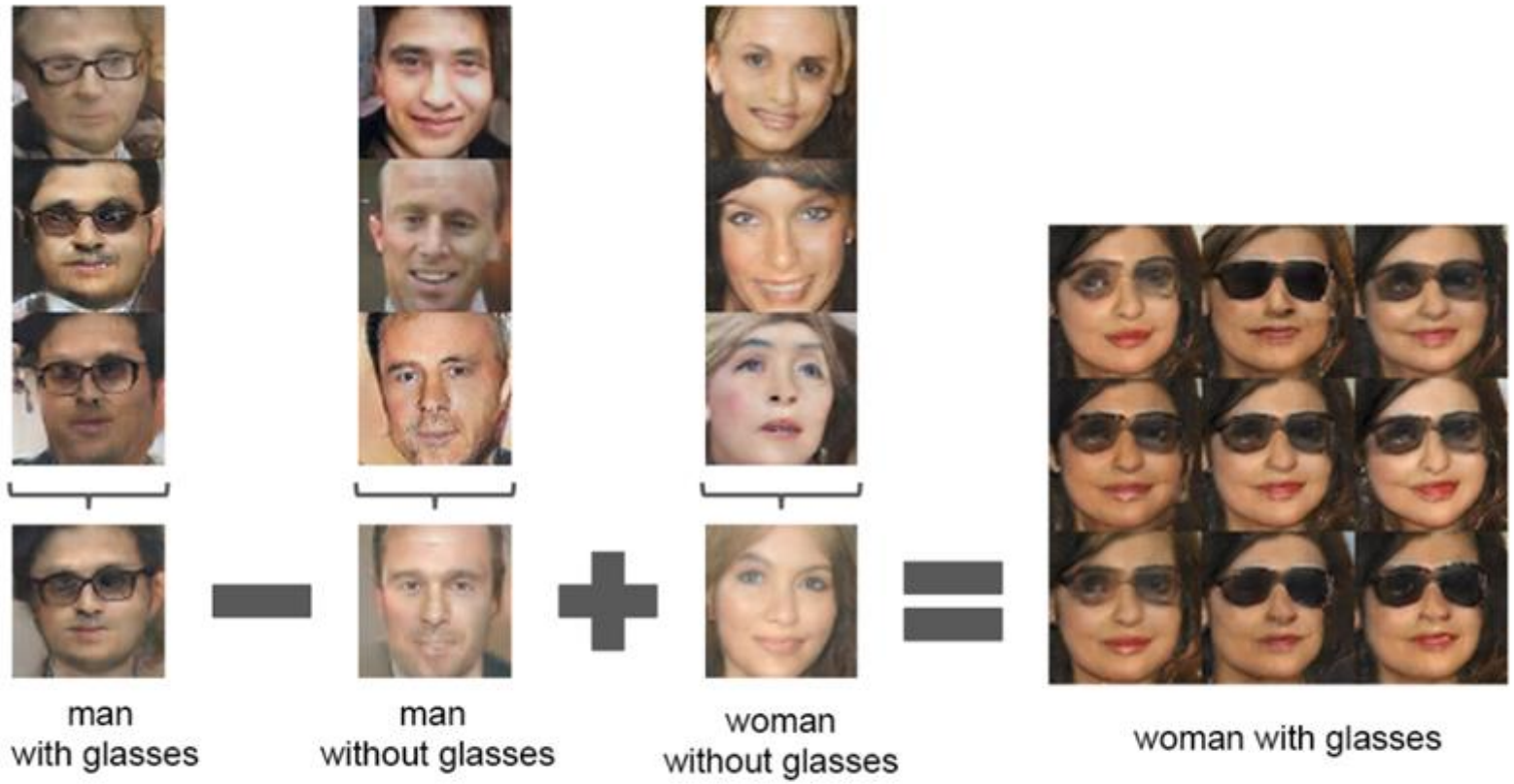
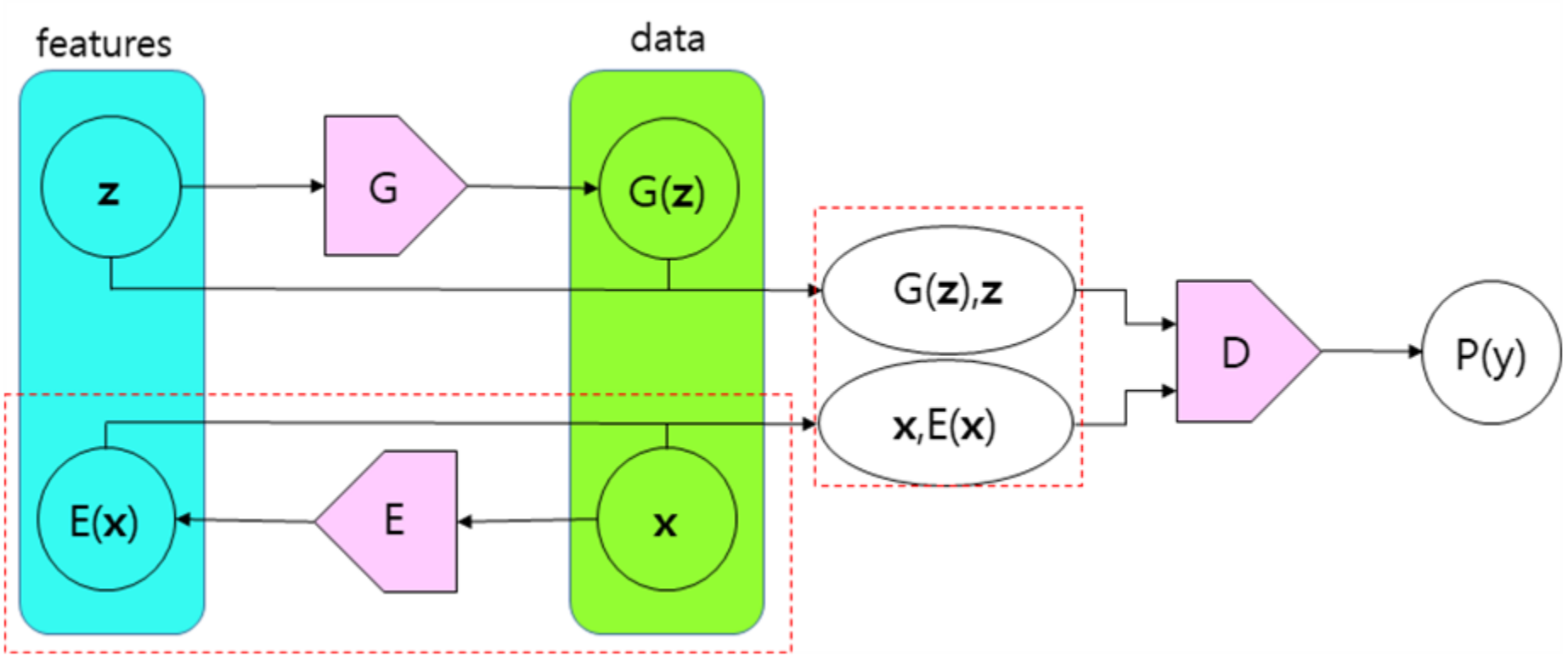


그림 13.4. 심층 컨볼루션 생성대립 회로망(DCGAN)을 얼굴 영상에 대하여 학습한 결과. 잡음 벡터의 연산에 의해 의미에 맞는 영상을 생성함 [A. Radford et al. 2016].

13.6. BiGAN



$$\min_{G,E} \max_{D} V(D,E,G) = E_{p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x}, E(\boldsymbol{x}))] + E_{p_z(z)}[\log(1 - D(G(z), z))]$$

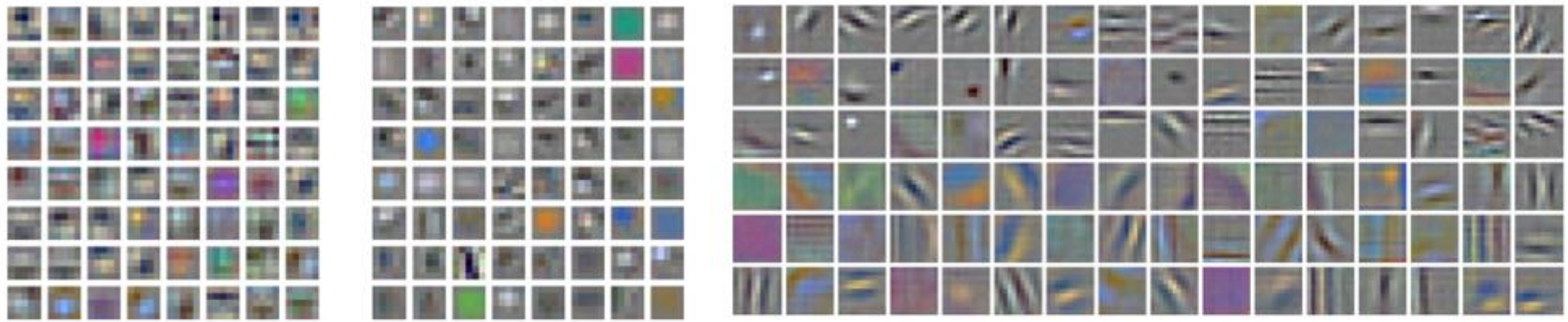


그림 13.6. BiGAN의 ImageNet 데이터 학습에 의해 얻어진 D, G, E의 컨볼루션 필터 [J. Donahue et al. 2017]

13.7 확산 모델 (Diffusion Models)

생성대립 회로망은 학습이 안정적이지 않고 대립망의 존재 때문에 생성된 출력의 다양성이 부족하다고 지적되고 있다. 확산모델은 비평형열역학(Non-equilibrium Thermodynamics)에 착안하여 대립망을 포함하지 않고 데이터를 생성하는 모델이다. 확산이란 물질이 고농도 지역에서 저농도 지역으로 이동하는 현상을 말한다. 물이 들어있는 잔에 잉크를 한 방울 떨어뜨렸다고 가정하자. 처음에는 한 지역에 고농도로 존재하는 잉크가 시간이 지날수록 확산되어 물 전체에 고르게 퍼지게 되고 결국 평형 상태에 이르게 될 것 이다. 이 과정을 역으로 재현하여 잉크가 물 전체에 퍼져있는 상태에서 처음에 잉크가 물에 떨어진 모양을 재현할 수 있을까? 확산모델은 이 과정을 역으로 재현하는 모델을 만들어내고자 한다.

확산모델은 주어진 데이터를 지속적으로 파괴하여 잡음만이 존재하게 되는 전방 확산과정(Forward Diffusion Process)과 잡음으로부터 데이터를 복원하는 역방향 확산과정(Reverse Diffusion Process)으로 이루어져 있다.

우선 전방확산과정을 자세히 알아보자. 학습데이터 집합 $\boldsymbol{X}_0$에서 데이터 $\boldsymbol{x}_0 \in \boldsymbol{X}_0$가 데이터 분포 $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$에 따라 샘플링되었다고 하자. 여기에 가우시안 잡음(Gaussian noise) $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$을 더하여

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\,\boldsymbol{\epsilon}, t = 1,2,\ldots,T \tag{13.7.1}$$

와 같이 표현된다고 하자. 이때, $\{\beta_t \in (0,1)\}_{t=1}^{T}$이다. 그러면, 확률분포 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$는 마코프 체인(Markov Chain)을 형성하여

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = N(\boldsymbol{x}_t; \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \;\; q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{13.7.2}$$

와 같이 주어지고, $\boldsymbol{x}_t (t = 1,2,\ldots,T)$는 $\boldsymbol{x}_0$와 같은 차원을 지닌다. 시간이 지날수록 가우시안 잡음에 의해 $\boldsymbol{x}_0$는 점점 그 특징을 잃어버리게 되고 마침내 $T \to \infty$에서 $\boldsymbol{x}_T$는 등방성 가우시안 분포(Isotopic Gaussina Distribution)를 지니게 된다.

식 (13.7.1)을 $\alpha_t = 1 - \beta_t$와 $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$로 변수를 치환하여 변환해보자. 먼저,

$$\boldsymbol{x}_1 = \sqrt{\alpha_1}\,\boldsymbol{x}_0 + \sqrt{1-\alpha_1}\,\boldsymbol{\epsilon} \tag{13.7.3}$$

$$\boldsymbol{x}_2 = \sqrt{\alpha_2}\,\boldsymbol{x}_1 + \sqrt{1-\alpha_2}\,\boldsymbol{\epsilon} = \sqrt{\alpha_2\alpha_1}\,\boldsymbol{x}_0 + \sqrt{1-\alpha_2\alpha_1}\,\boldsymbol{\epsilon} \tag{13.7.4}$$

와 같다. 여기서, 두 개의 가우시안 변수 $N(\mathbf{0}, \sigma_1^2\boldsymbol{I})$와 $N(\mathbf{0}, \sigma_2^2\boldsymbol{I})$를 더하면 $N(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\boldsymbol{I})$가 되는 성질을 이용하였다. 마찬가지로

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\,\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon} = \sqrt{\overline{\alpha_t}}\,\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\,\boldsymbol{\epsilon} \tag{13.7.5}$$

를 얻게 된다. 즉, $\boldsymbol{x}_t\,(t = 1, 2, \ldots, T)$가 $\boldsymbol{x}_0$와 $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$ 만으로 표현되므로 전방확산과정은 학습이 필요 없다.

이제 역방향확산과정을 알아보자. $\boldsymbol{x}_T \sim N(\mathbf{0}, \boldsymbol{I})$인 가우시안 잡음의 샘플로부터 조건확률분포 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$에 따라 변환을 하면 $\boldsymbol{x}_0$를 재현할 수 있을 것이다. 그렇지만, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$를 쉽게 알아낼 수 없기 때문에, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$를 근사화하기 위하여

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{13.7.6}$$

를 학습하도록 하자. 그림 13.7은 이러한 개념을 보여준다.

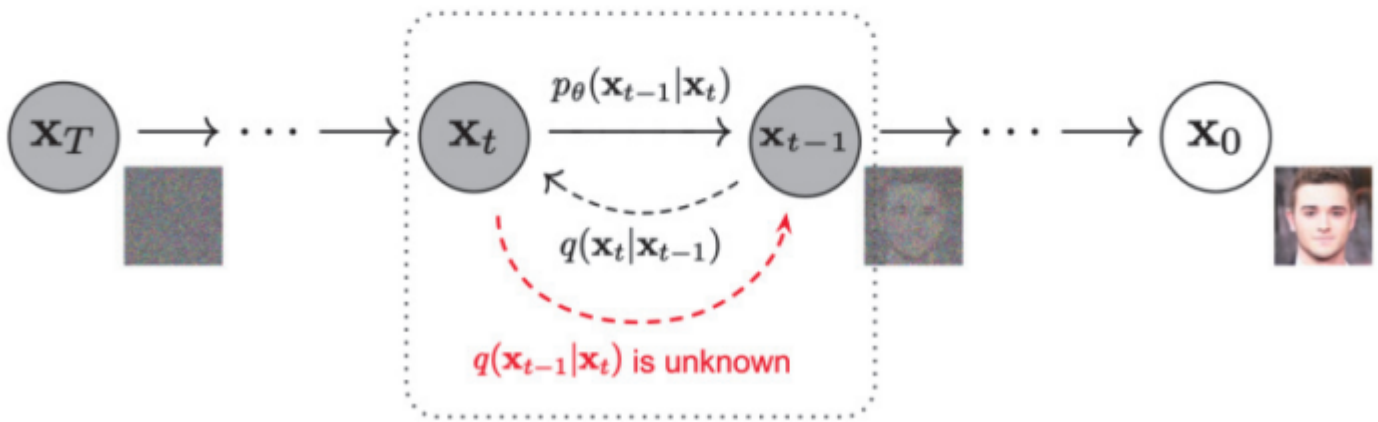


그림 13.7. 전방확산과정과 역방향확산과정 개념도[17]

학습은 역방향확산과정의 조건확률분포 $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = N(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t),\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t,t))$를 신경회로망으로 근사화하는 것이다. 따라서, 학습 기준 L_t는 $\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t,\boldsymbol{x}_0)$ 와 $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ 사이의 차이를 최소화 시키는 형태로

$$L_t = E_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t,t)\|_2^2}\|\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t,\boldsymbol{x}_0)-\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t)\|^2\right] \tag{13.7.17}$$

와 같이 표현할 수 있다. 보통 역방향확산모델의 구현은 U-net을 사용한다.