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 $p(\boldsymbol{x})$

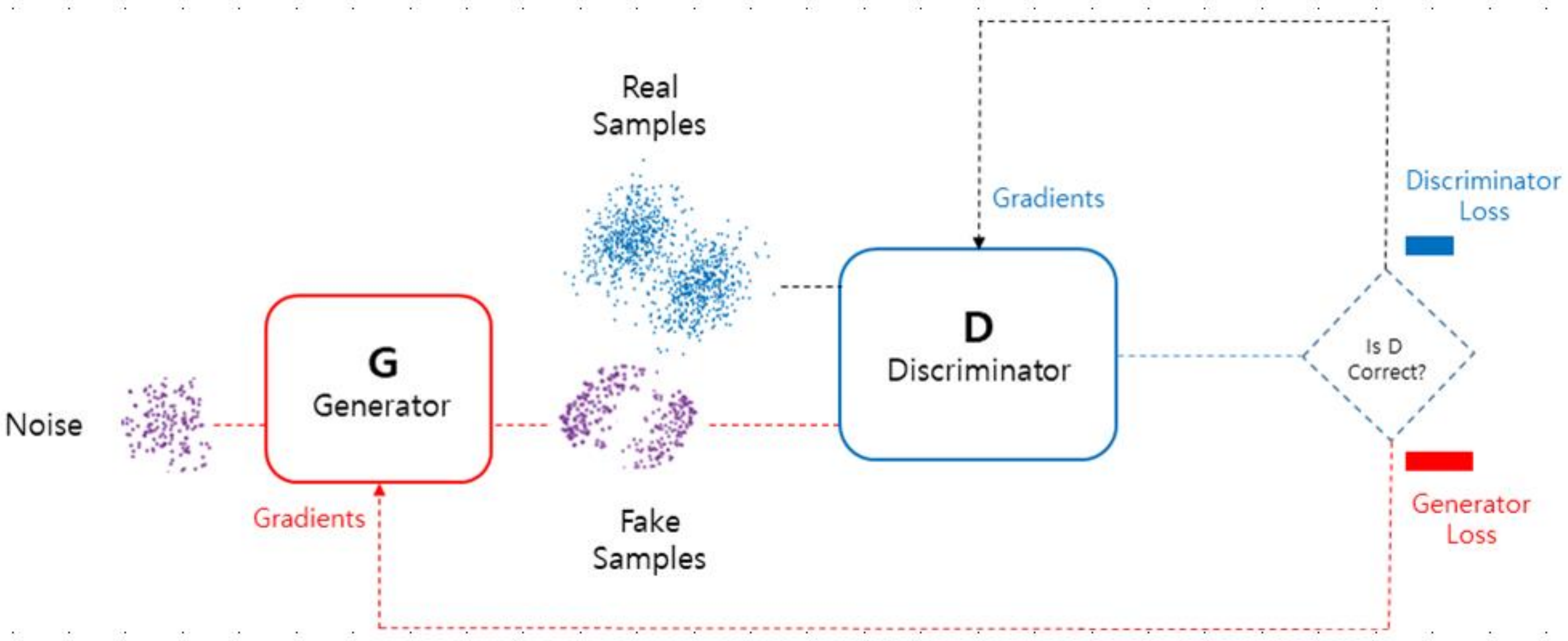
- Discriminator (supervised learning)

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

- 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.

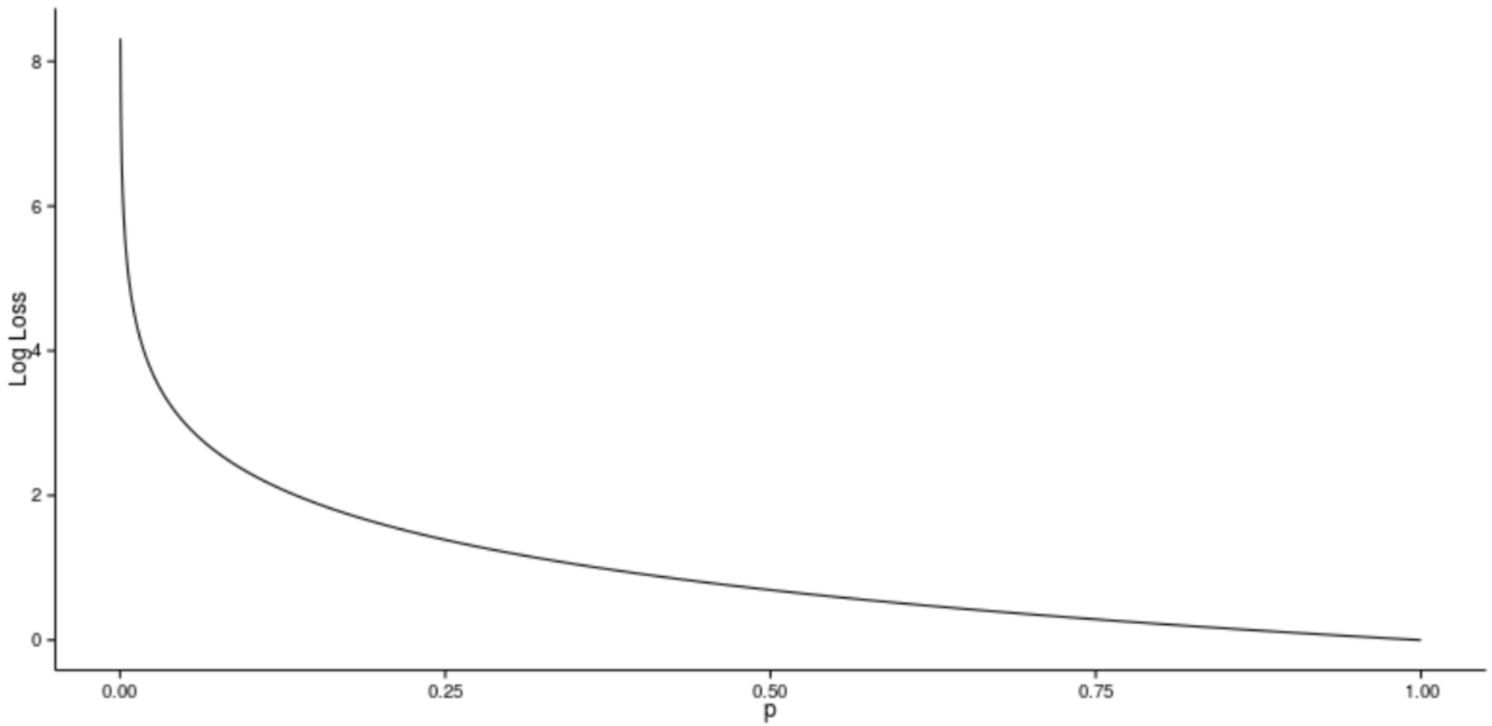
13.3. Mathematical Modelling of GAN

- 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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

  - 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

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 $V(D,G)$

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

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□

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

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

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

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.

13.4. Coding a GAN

- Leaky ReLU

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

- Training Discriminator
  - Maximizing

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

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

- Training a Generator
  - Minimizing

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

13.5. DCGAN

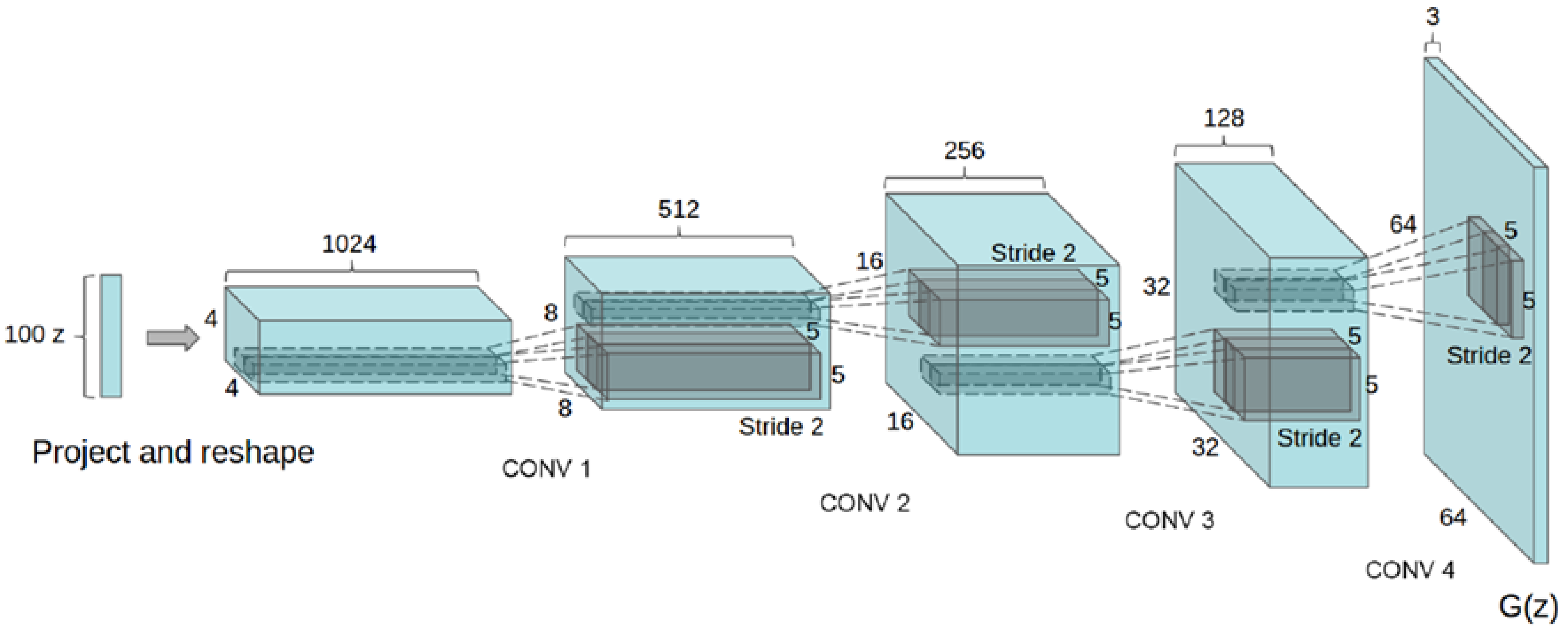


그림 13.2. 심층 컨볼루션 생성대립 회로망(DCGAN)의 생성기 모델. 100차원의 균일분포 잡음벡터 입력에 대하여 4층의 이동폭 2인 이항 컨볼루션 필터가 적용되어 64×64 영상을 생성함 [A. Radford et al. 2016].

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

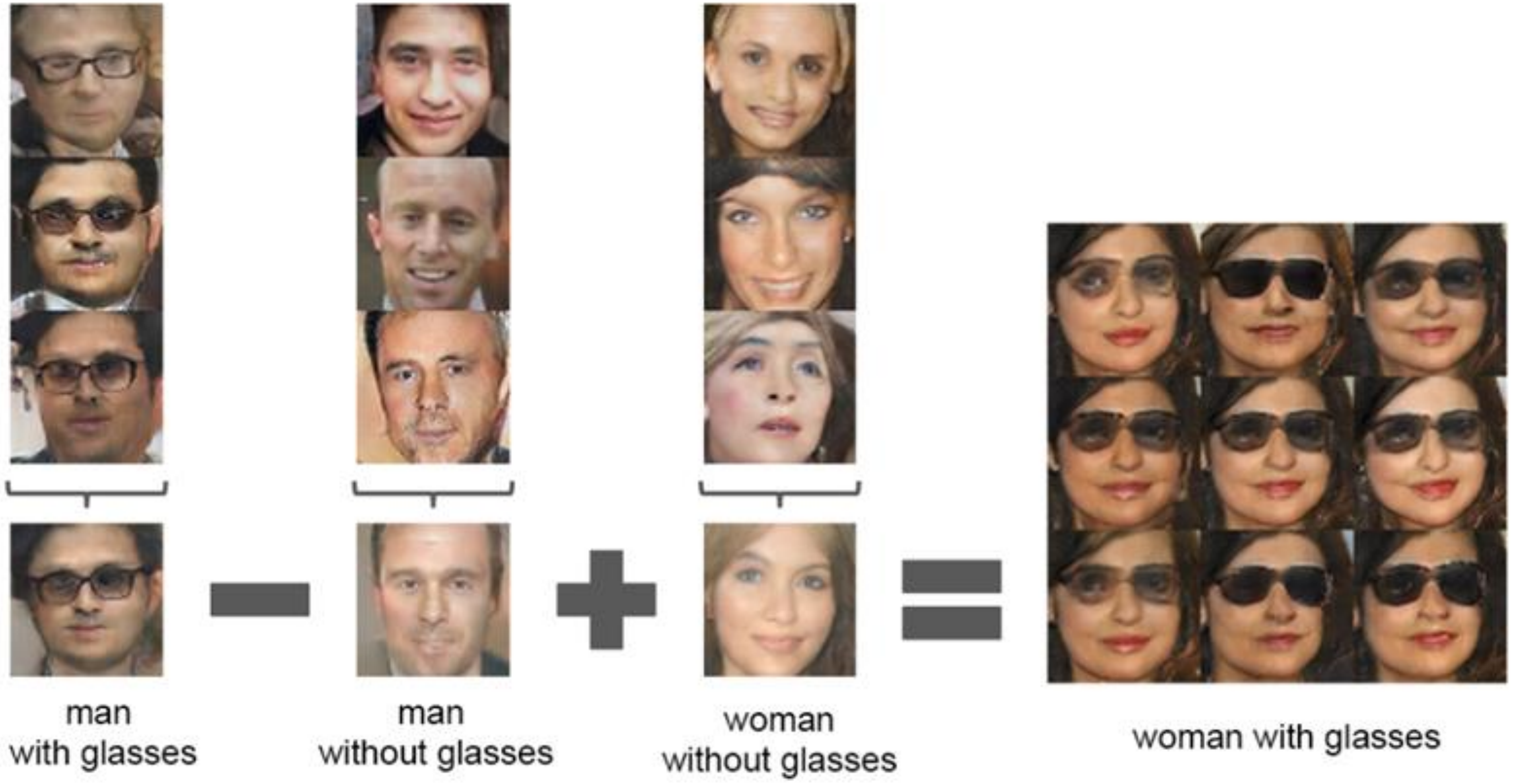
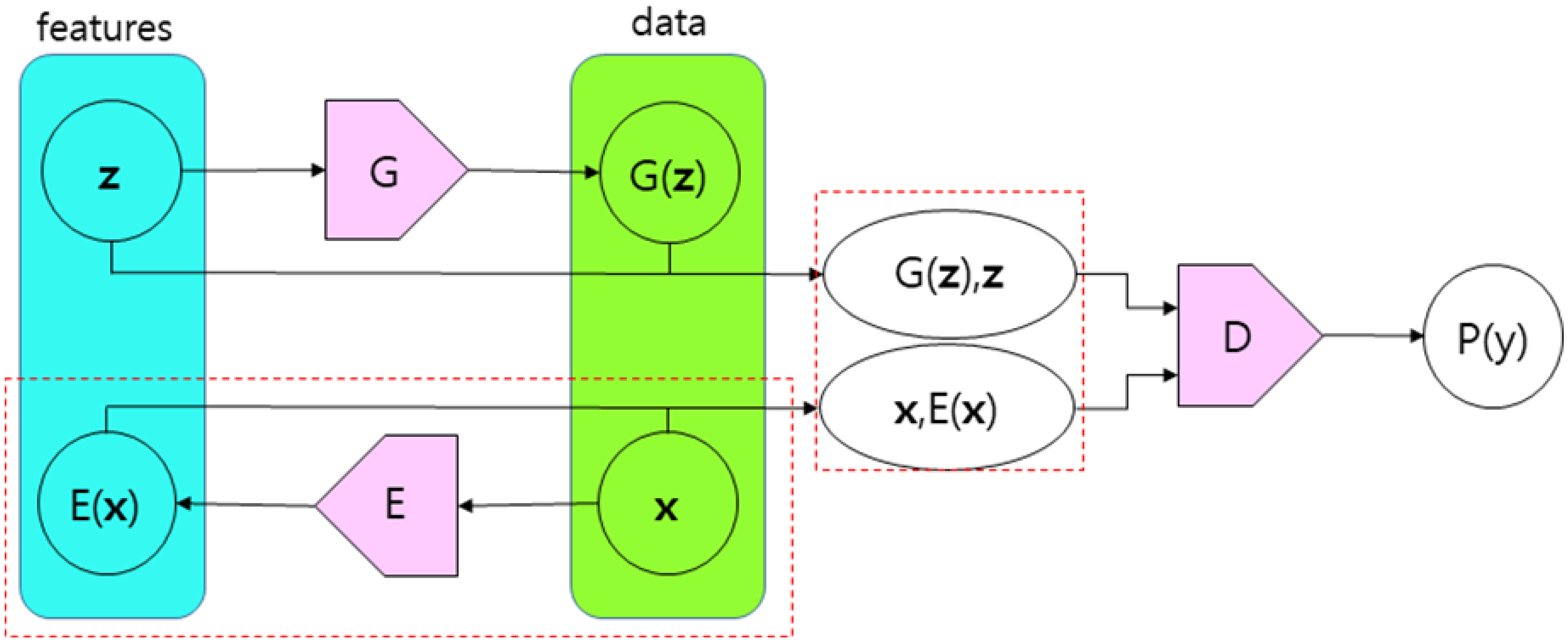


그림 13.4. 심층 컨볼루션 생성대립 회로망(DCGAN)을 얼굴 영상에 대하여 학습한 결과. 잡음 벡터의 연산에 의해 의미에 맞는 영상을 생성함 [A. Radford et al. 2016].

13.6. BiGAN



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

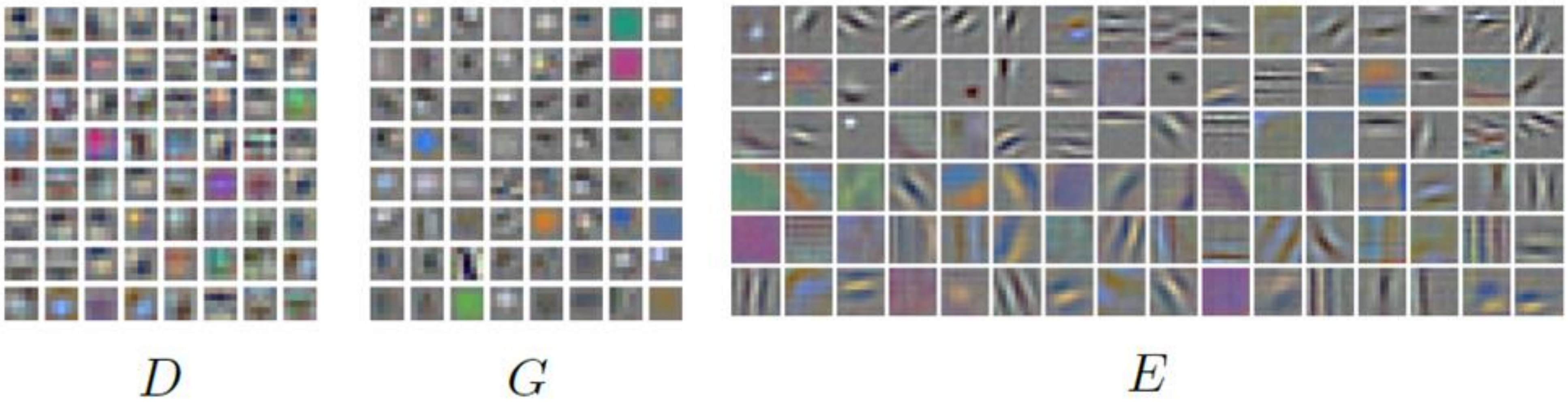


그림 13.6. BiGAN의 ImageNet 데이터 학습에 의해 얻어진 D, G, E의 컨볼루션 필터 [J. Donahue et al. 2017]