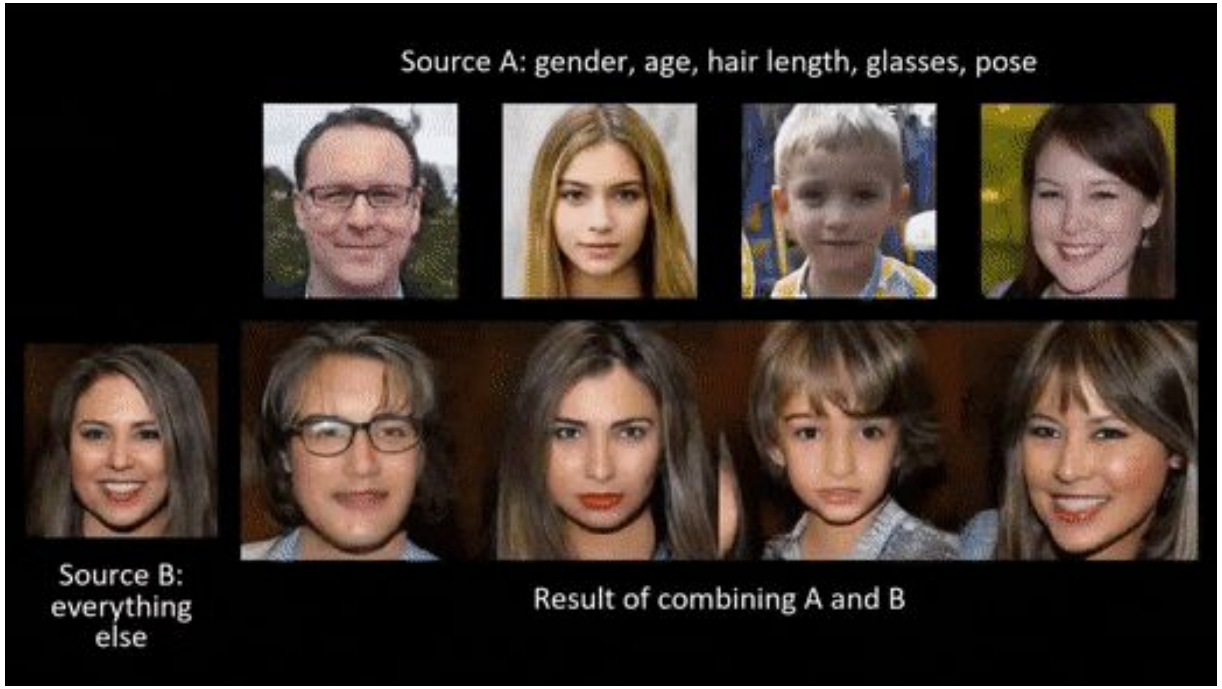
Генеративно змагальні мережі

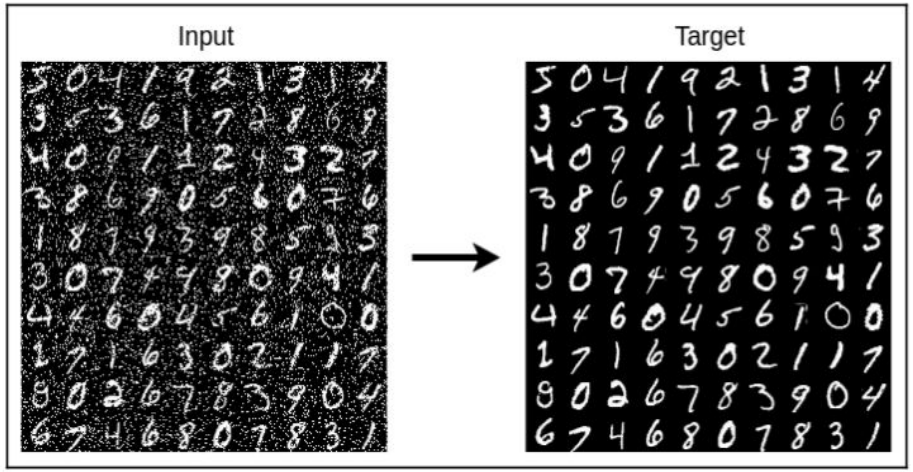


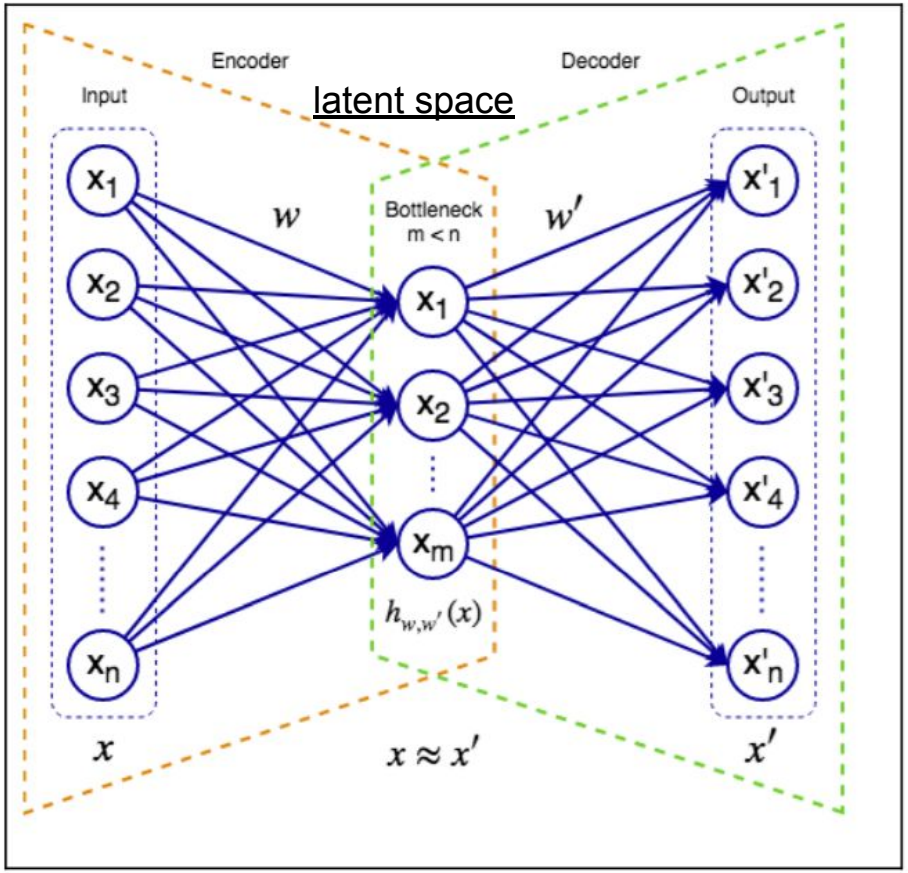
Автоенкодер

reconstruction error $\mathcal{L} = (x, x')$

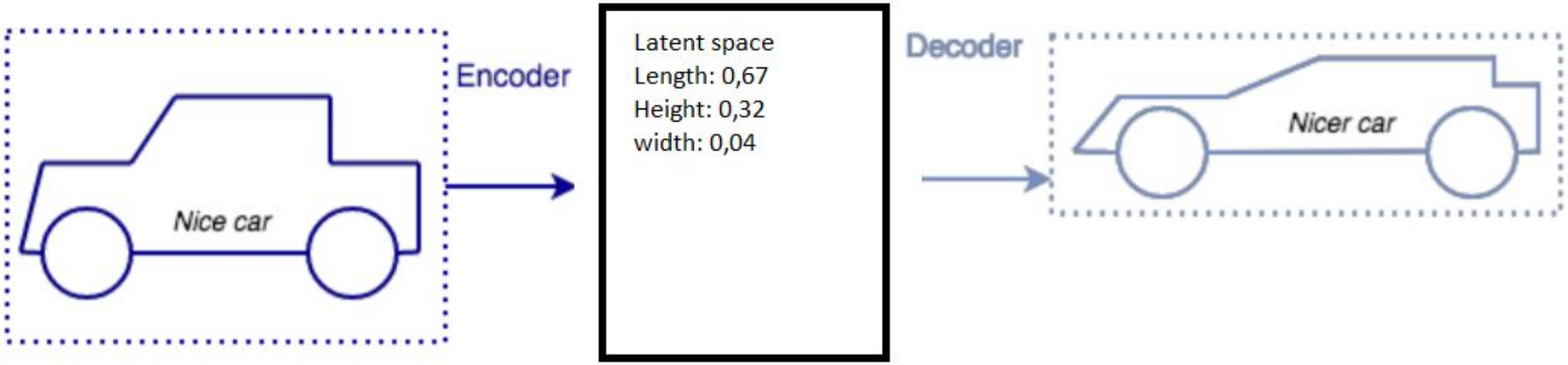
a compact representation of the data

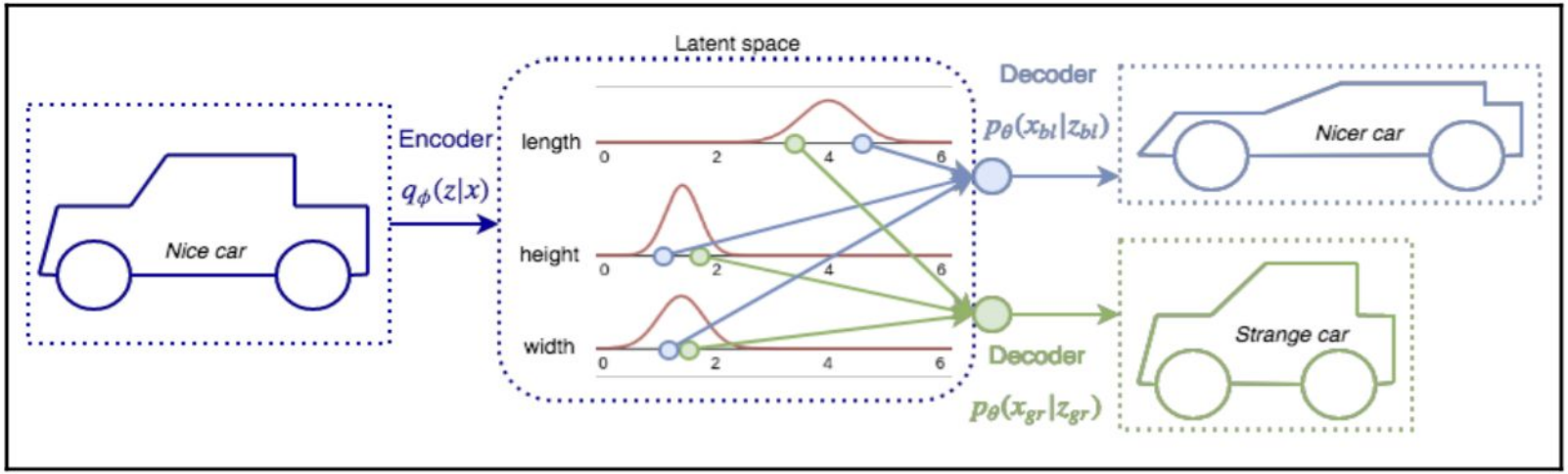
the most important features





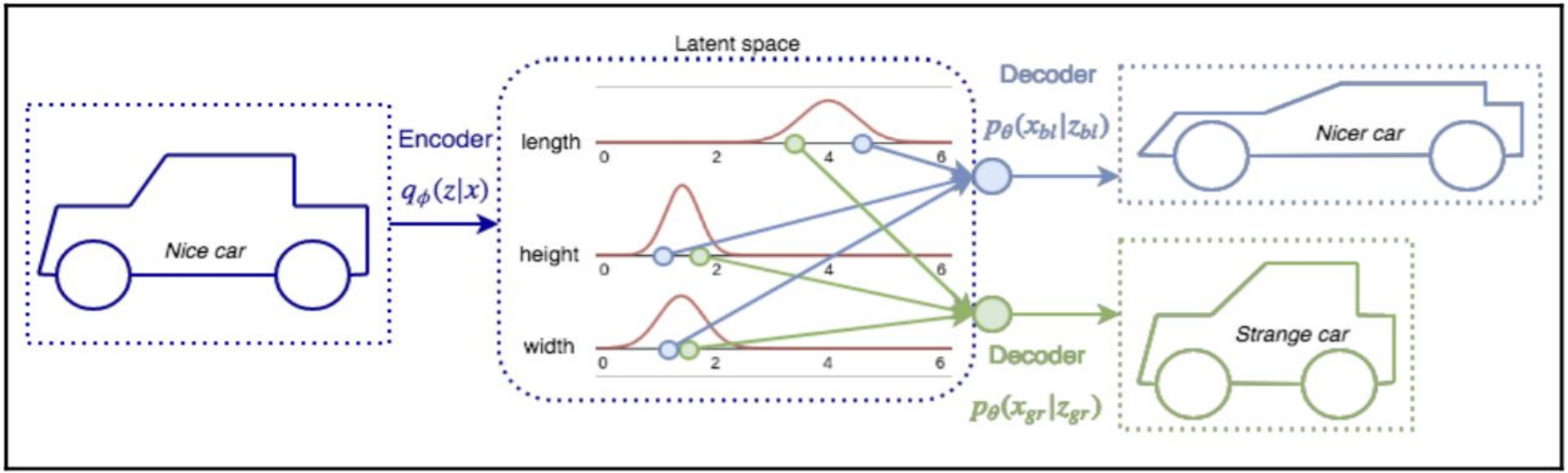
Варіаційний автоенкодер





1. Φ - ваги і зсув мережі енкодера; x - вхід; z - представлення в прихованому просторі. Вихід автоенкодера це розподіл (наприклад гаусівський) за можливими значеннями z, які могли б створити x.
2. Θ - ваги і зсув мережі декодера; z вибирається випадковим чином з розподілу, потім передається через декодер, вихідним результатом якого є розподіл на можливі відповідні значення x.

$$L(\theta, \varphi; x) = -D_{KL}(q_\varphi(z|x)||p_\theta(z)) + E_{q_\varphi(z|x)}[\log(p_\theta(x|z))]$$

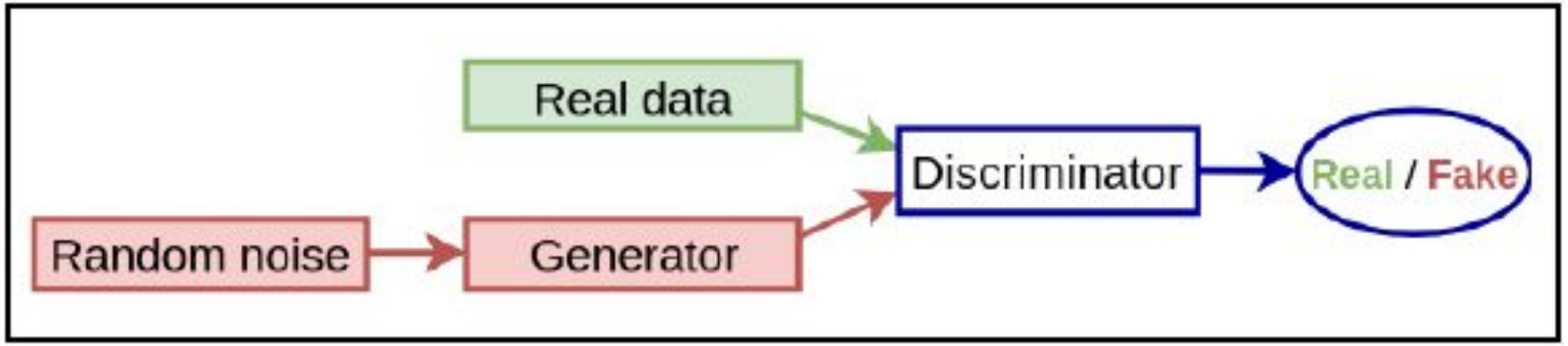


В автоенкодері можуть використовуватися лише повнозв'язні згорткові мережі?

Так

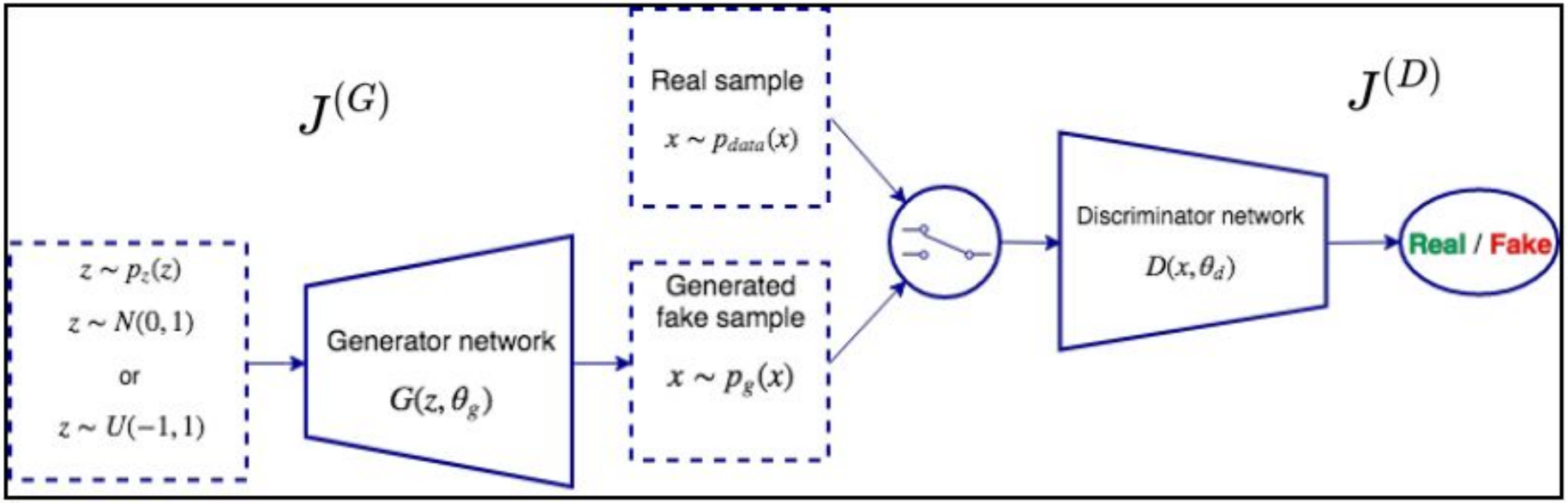
Ні

Generative Adversarial networks



Генератор: намагається створити реалістичне вихідне зображення

Дискримінатор: намагається визначити, чи вхідне зображення походить від реальних зображень чи з генерованих.



Послідовна мінімаксна гра з нульовою сумою

Послідовна - дискримінатор і генератор мінімізують свої ц.ф. по черзі

Нульова сума - $J^{(G)} = -J^{(D)}$

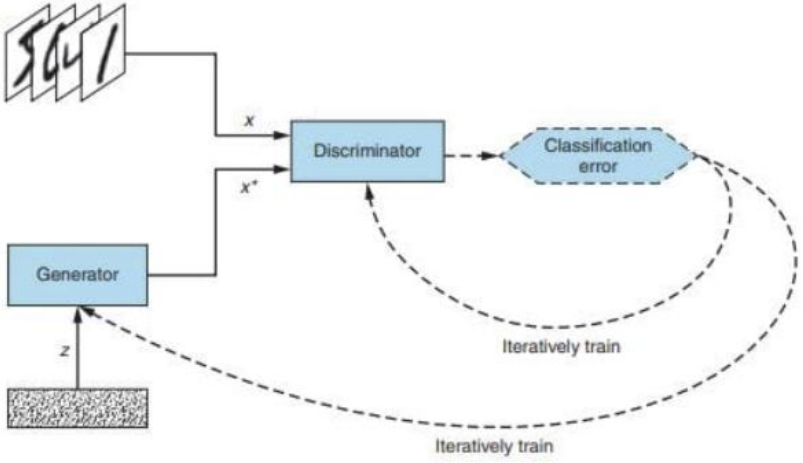
Мінімаксна - $\min_G \max_D V(G, D)$

Навчання дискримінатора

1. Залежно від вхідної вибірки (реальної чи підробленої), маємо:
- Вибрати зразок із реальних даних $x \sim p_{data}$ та використайте його для отримання D(x)
- Створити підроблений зразок, $x \sim p_g$. $z \rightarrow G(z) \rightarrow D(G(z))$

2. Обчисліть функцію втрат, яка відображає подвійність даних тренувань.

3. Зворотне розповсюдження градієнта помилок та оновлення ваг.



Функція втрат

$$H(p,q) = -\sum_{i=1}^{n} p_i(x)\log(q_i(x))$$

$$H(p,q) = -(p(x)\log q(x) + (1-p(x))\log(1-q(x)))$$

$$H(p,q) = -\frac{1}{m}\sum_{j=1}^{m}(p(x_j)\log(q(x_j)) + (1-p(x_j))\log(1-q(x_j)))$$

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}}\log(D(x)) - \frac{1}{2}\mathbb{E}_z\log(1-D(G(z)))$$

Навчання генератора

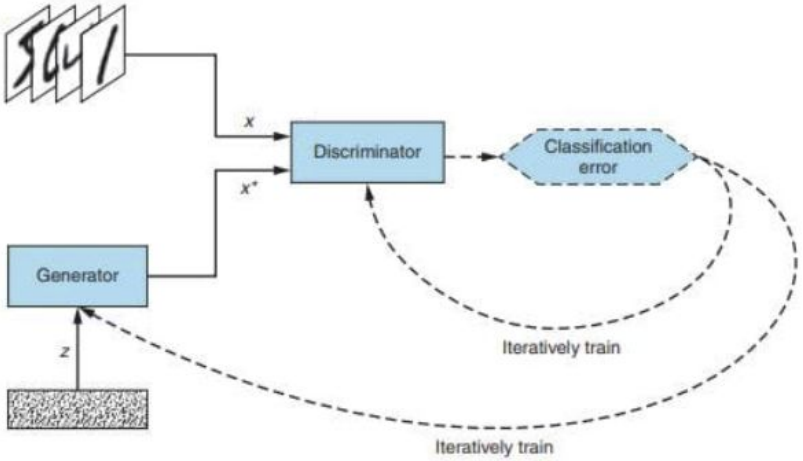
1. Починаємо з випадкового прихованого вектора z, і подаємо його через генератор і дискримінатор, щоб отримати результат, d(g(z)).
2. Функція втрат така ж, як і втрати дискримінатора.
3. У зворотному проході θd, зафіксовані, і регулюємо θg.

$$J^{(G)} = \mathbb{E}_z \log(1 - D(G(z)))$$

$$J^{(G)} = -\mathbb{E}_z \log(D(G(z))$$

$$\min_G \max_D V(G,D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log(D(x)) + \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z)))$$

- Генератор намагається мінімізувати ціль, тоді як дискримінатор намагається максимізувати її.



Алгоритм навчання

- Повторіть для Ф ітерацій:

1. Повторіть для k кроків, де k - гіперпараметр:

- Виберіть міні-батч з m випадкових зразків із прихованого простору,

$$\{z^{(1)}, z^{(2)}, \dots z^{(m)}\} \sim p_g(z).$$

- Виберіть міні-партію зразків з реальних даних,

$$\{x^{(1)}, x^{(2)}, \dots x^{(m)}\} \sim p_{data}(x).$$

- Оновіть ваги дискримінатора θd, у напрямку росту стохастичного градієнта функції втрат:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))]$$

2. Виберіть міні-батч з m випадкових зразків із прихованого простору,

$$\{z^{(1)}, z^{(2)}, \dots z^{(m)}\} \sim p_g(z).$$

3. Оновіть генератор, у напрямку зменшення стохастичного градієнт функції втрат:

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

Яке з тверджень найбільше відповідає дійсності?

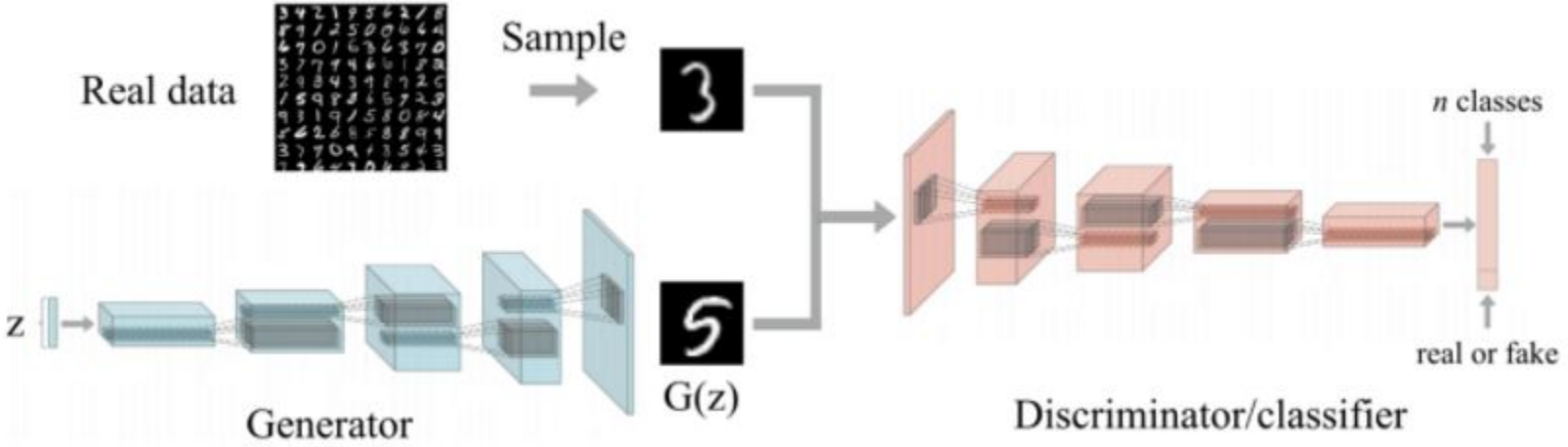
милка дискримінатора дорівнює помилці генератора

скримінатор та генератор корегують свої ваги почерзі

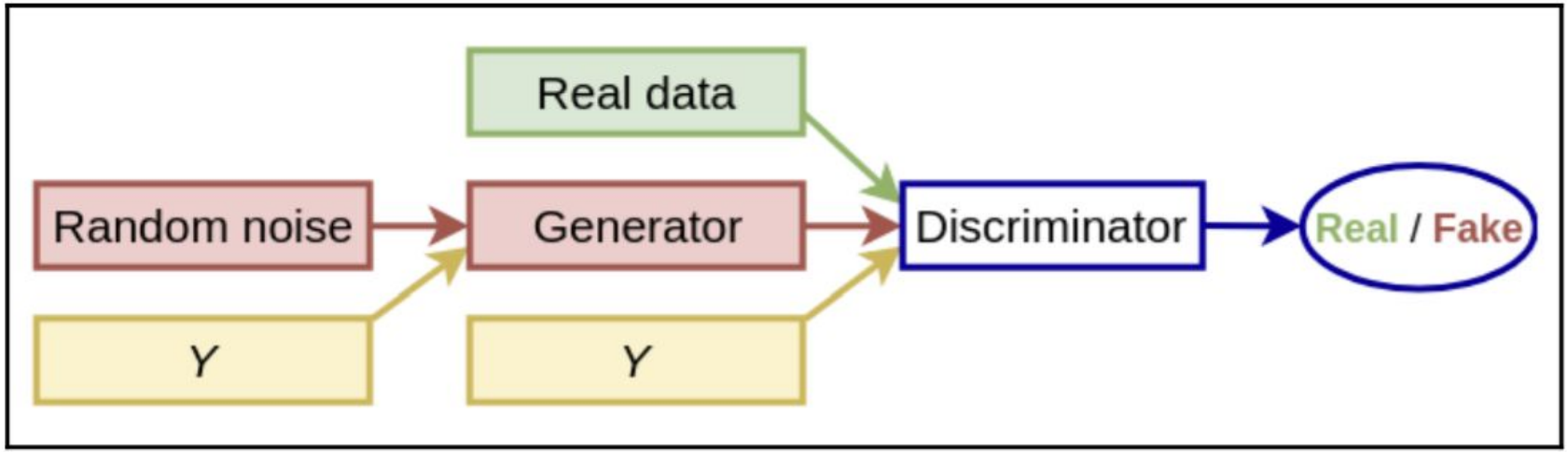
вага між генератором і дискримінатором встановлюється коли генерується одне найбільш реалістичне зображення

Deep Convolutional Generative Adversarial networks (DCGANs)

- Дискримінатор використовує згортку зі страйдом замість пулінгу.
- Генератор використовує транспоновану згортку.
- Використовується батч нормалізація.
- ReLU, LeakyReLu
- Немає повнозв’язних шарів



Conditional GAN (cGAN)



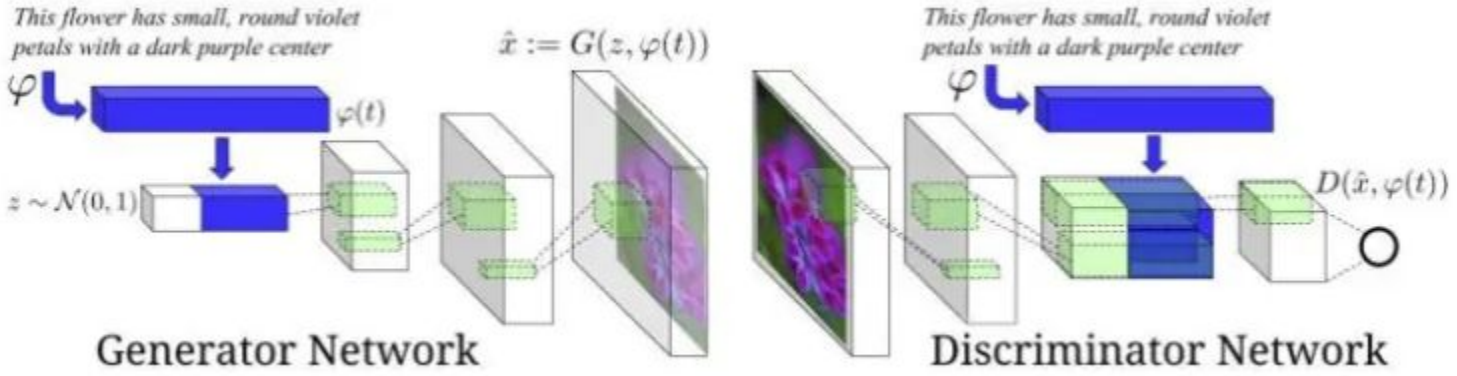
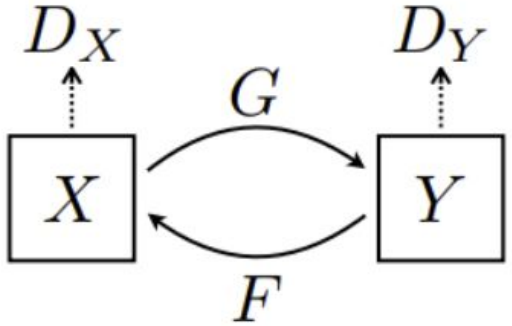


Image-to-Image Translation and CycleGAN

There are 2 generators (G and F) and 2 discriminators (X and Y) being trained here.

- Generator G learns to transform image X to image Y. (G:X−>Y)
- Generator F learns to transform image Y to image X. (F:Y−>X)
- Discriminator D_X learns to differentiate between image X and generated image X (F(Y)).
- Discriminator D_Y learns to differentiate between image Y and generated image Y (G(X)).

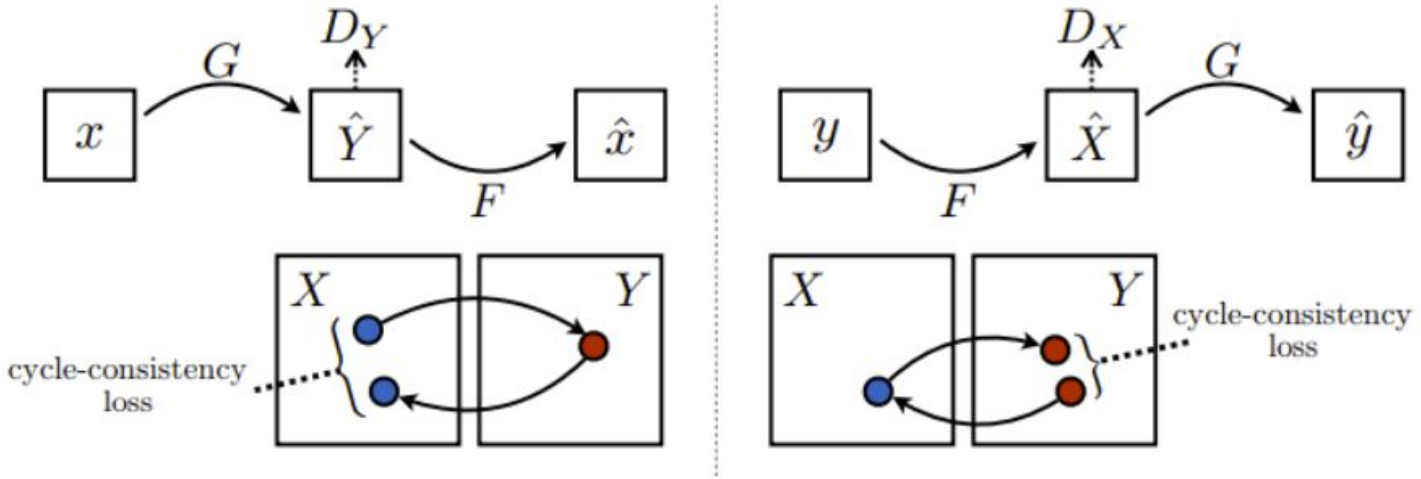


In cycle consistency loss,

- Image X is passed via generator G that yields generated image $\hat{Y}$.
- Generated image $\hat{Y}$ is passed via generator F that yields cycled image $\hat{X}$.
- Mean absolute error is calculated between X and $\hat{X}$.

$$forward\ cycle\ consistency\ loss : X -> G(X) -> F(G(X)) \sim \hat{X}$$

$$backward\ cycle\ consistency\ loss : Y -> F(Y) -> G(F(Y)) \sim \hat{Y}$$



Хто вже використовував(працював) GAN?

Я

Не я

https://www.tensorflow.org/tutorials/generative/cvae