

**Generowanie cyfr pisanych odręcznie na bazie sieci  
neuronowej o architekturze autoencodera  
Analiza i implementacja różnych modeli generatywnych**

Prowadzący: Adam Świtoński

Politechnika Śląska

Maj 2025

# Plan prezentacji

- 1 Modele
- 2 Porównanie modeli
- 3 Wyzwania implementacyjne
- 4 Przykłady zastosowań

### Cel projektu

Zastosowanie sieci neuronowej o architekturze autoencodera do generowania niskorozdzielczych obrazów cyfr pisanych odręcznie.

- Trenowanie różnych wariantów autoencoder'a
- Generowanie nowych cyfr poprzez podawanie losowych wartości na wejście dekodera
- Badanie wpływu struktury sieci (liczba warstw, liczba neuronów) oraz parametrów uczenia
- Wykorzystanie zbioru danych MNIST

### Rozważane warianty

Klasyczny autoencoder, wariacyjny autoencoder (VAE), GAN, Diffusion, VQ-VAE, Conditional VAE

## Autoencoder

**Autoencoder** to rodzaj sieci neuronowej, która uczy się kompresować dane wejściowe do reprezentacji o niższym wymiarze (kod), a następnie rekonstruować oryginalne dane z tej reprezentacji.

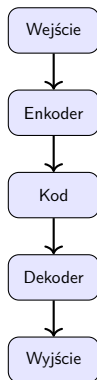
### Zastosowania:

- Redukcja wymiarowości
- Denoising (odszumianie)
- Generowanie nowych danych

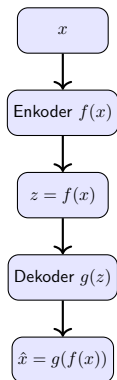
**Zalety:** prostota, szybki trening, interpretowalność

**Wady:** ograniczona zdolność generatywna, brak kontroli nad rozkładem latentnym

**Bibliografia:** [2]



## Autoencoder - Funkcja straty (1/2)



### Struktura Autoencodera:

Autoencoder składa się z enkodera, który kompresuje dane wejściowe do reprezentacji latentnej, oraz dekodera, który rekonstruuje dane z tej reprezentacji.

Kluczowe elementy to dane wejściowe  $x$ , reprezentacja latentna  $z$ , oraz rekonstrukcja  $\hat{x}$ .

## Autoencoder - Funkcja straty (2/2)

$$\begin{aligned}\mathcal{L}_{AE} &= \|x - \hat{x}\|^2 \\ &= \|x - g(f(x))\|^2\end{aligned}$$

**Objaśnienie funkcji straty:** gdzie:

- $x$  - dane wejściowe (obraz)
- $f(x)$  - funkcja enkodera
- $z = f(x)$  - reprezentacja latentna
- $g(z)$  - funkcja dekodera
- $\hat{x} = g(f(x))$  - rekonstrukcja

Funkcja straty to typowo błąd średniokwadratowy (MSE) lub binary cross-entropy dla obrazów.

## Wariacyjny Autoencoder (VAE)

**VAE** to probabilistyczne rozszerzenie autoencodera, które modeluje rozkład latentny danych.

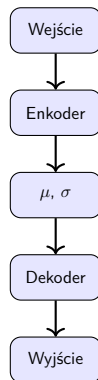
### Kluczowe cechy:

- Modelowanie rozkładu latentnego ( $\mu$ ,  $\sigma$ )
- Regularyzacja poprzez KL-dywersję
- Generowanie nowych próbek przez próbkowanie

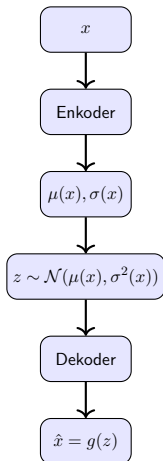
**Zalety:** generatywność, ciągła przestrzeń latentna, możliwość interpolacji

**Wady:** rozmyte próbki, trudność w trenowaniu

**Bibliografia:** [4]



## Wariacyjny Autoencoder (VAE) - Funkcja straty (1/2)



### Struktura VAE: Wariacyjny

Autoencoder modeluje rozkład latentny poprzez parametry  $\mu$  i  $\sigma$ , co pozwala na próbkowanie  $z$  z rozkładu normalnego.

Proces obejmuje kodowanie danych wejściowych  $x$  do przestrzeni latentnej oraz dekodowanie do rekonstrukcji  $\hat{x}$ .



## Wariacyjny Autoencoder (VAE) - Funkcja straty (2/2)

$$\mathcal{L}_{VAE} = \underbrace{\|x - \hat{x}\|^2}_{\text{Rekonstrukcja}} + \underbrace{\beta \cdot D_{KL}(q(z|x) \| p(z))}_{\text{Regularyzacja KL}}$$

**Objaśnienie funkcji straty:** gdzie:

- $q(z|x) = \mathcal{N}(z; \mu(x), \sigma^2(x))$  - kodowanie
- $p(z) = \mathcal{N}(z; 0, I)$  - rozkład a priori
- $\beta$  - waga regularyzacji KL

Dla rozkładu normalnego, dywergencja KL wynosi:

$$D_{KL} = \frac{1}{2} \sum_{j=1}^d (1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2)$$

## Conditional VAE

**Conditional VAE** (CVAE) to wariacyjny autoencoder, który dodatkowo warunkuje generowanie na zadanej klasie (np. cyfra).

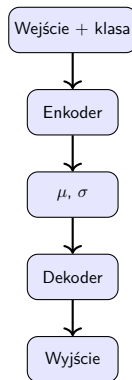
### Kluczowe cechy:

- Warunkowanie generowania na etykietach
- Możliwość sterowania procesem generacji
- Łączenie etykiet z danymi wejściowymi

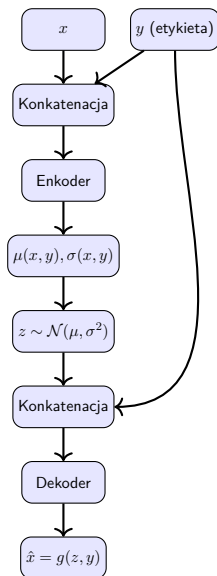
**Zalety:** kontrola nad generowanymi danymi, elastyczność

**Wady:** większa złożoność, wymaga etykiet

**Bibliografia:** [6]



## Conditional VAE - Funkcja straty (1/2)



### Struktura Conditional VAE:

Conditional VAE uwzględnia etykietę  $y$  (np. klasę cyfry) podczas kodowania i dekodowania, co pozwala na kontrolowanie generowanych danych.

Etykieta jest konkatelowana zarówno z danymi wejściowymi, jak i z próbkowaną reprezentacją latentną  $z$ .

## Conditional VAE - Funkcja straty (2/2)

$$\mathcal{L}_{CVAE} = \underbrace{\|x - \hat{x}\|^2}_{\text{Rekonstrukcja}} + \underbrace{\beta \cdot D_{KL}(q(z|x, y) \| p(z|y))}_{\text{Regularyzacja KL}}$$

**Objaśnienie funkcji straty:** gdzie:

- $q(z|x, y) = \mathcal{N}(z; \mu(x, y), \sigma^2(x, y))$
- $p(z|y)$  - warunkowy rozkład a priori
- $y$  - etykieta klasy (np. cyfra 0-9)

Podczas generowania:

- Wybierz etykietę  $y$  (np. "3")
- Próbkuj  $z \sim \mathcal{N}(0, I)$
- Wygeneruj  $\hat{x} = g(z, y)$

## VQ-VAE

**VQ-VAE** to autoencoder, w którym przestrzeń latentna jest kwantyzowana do skończonego zbioru wektorów (słownik kodów).

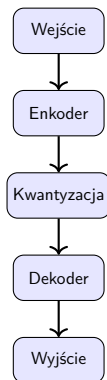
### Kluczowe cechy:

- Dyskretna przestrzeń latentna
- Kwantyzacja wektorowa
- Słownik kodowy

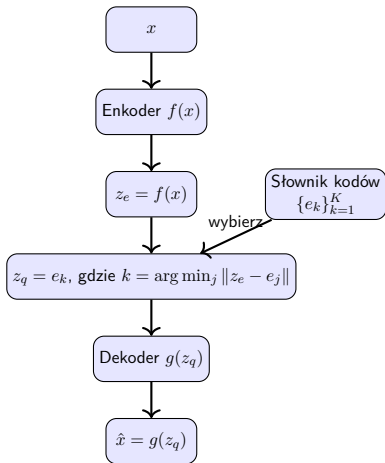
**Zalety:** dyskretna reprezentacja, dobre wyniki w generowaniu sekwencji

**Wady:** trudność w trenowaniu, konieczność doboru rozmiaru słownika

**Bibliografia:** [5]



## VQ-VAE - Funkcja straty (1/2)



### Struktura VQ-VAE: VQ-VAE

kwantyzuje wyjście enkodera  $z_e$  do najbliższego wektora  $z_q$  ze słownika kodów, co prowadzi do dyskretnej reprezentacji latentnej.

Słownik kodów jest kluczowym elementem, który pozwala na wybór odpowiedniego wektora podczas procesu kwantyzacji.

## VQ-VAE - Funkcja straty (2/2)

**Objaśnienie funkcji straty:** gdzie:

- $z_e = f(x)$  - wyjście enkodera
- $e$  - wybrany wektor ze słownika
- $z_q$  - kwantyzowany wektor
- $sg[]$  - operacja "stop gradient"
- $\beta$  - współczynnik commitment loss

Słownik kodów jest aktualizowany poprzez:

$$e_j = \text{MA}(z_e), \text{ dla } j = \arg \min_k \|z_e - e_k\|$$

gdzie MA to średnia krocząca.

$$\mathcal{L}_{VQ-VAE} = \underbrace{\|x - \hat{x}\|^2}_{\text{Rekonstrukcja}} + \underbrace{\|sg[z_e] - e\|^2}_{\text{Uaktualnienie słownika}} + \underbrace{\beta \|z_e - sg[e]\|^2}_{\text{Commitment loss}}$$

# Generative Adversarial Network (GAN)

**GAN** to model generatywny składający się z dwóch sieci: generatora (tworzy próbki) i dyskryminatora (odróżnia próbki prawdziwe od fałszywych).

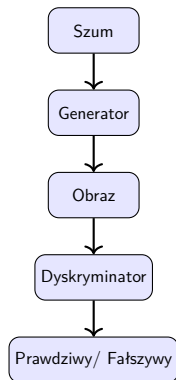
## Kluczowe cechy:

- Układ rywalizujący (gra dwuosobowa)
- Generator tworzy coraz lepsze próbki
- Dyskryminator staje się coraz trudniejszy do oszukania

**Zalety:** realistyczne próbki, duża elastyczność

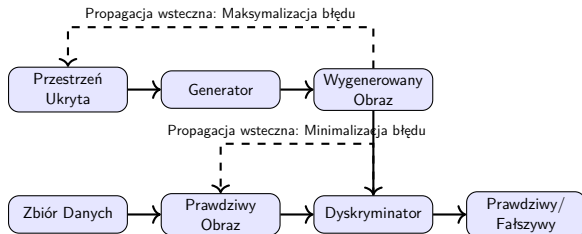
**Wady:** trudność w trenowaniu, niestabilność, mode collapse

**Bibliografia:** [1]





## GAN - Funkcja straty (1/2)



### Struktura GAN:

Składa się z dwóch sieci neuronowych rywalizujących ze sobą:

- **Generator** - tworzy obrazy z losowego szumu (przestrzeń ukryta), starając się oszukać dyskryminator
- **Dyskryminator** - ocenia, czy obraz jest prawdziwy (z zbioru danych) czy wygenerowany

Podczas treningu:

- Generator maksymalizuje błąd dyskryminatora
- Dyskryminator minimalizuje własny błąd

Ta rywalizacja stopniowo poprawia jakość generowanych obrazów.

## GAN - Funkcja straty (2/2)

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

**Objaśnienie funkcji straty:**

**Trening dyskryminatora  $D$ :**

$$\max_D \mathbb{E}_x [\log D(x)] + \mathbb{E}_z [\log(1 - D(G(z)))]$$

**Trening generatora  $G$ :**

$$\min_G \mathbb{E}_z [\log(1 - D(G(z)))]$$

**Proces treningu:**

- Naprzemienne aktualizacje  $D$  i  $G$
- Balansowanie jakości i różnorodności

## Diffusion Model

**Model dyfuzji** to nowoczesny model generatywny, który uczy się odsumiania danych przez odwracanie procesu stopniowego dodawania szumu.

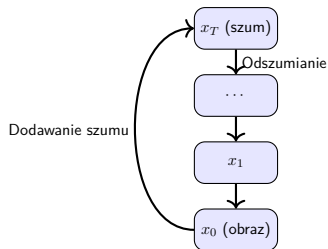
### Kluczowe cechy:

- Proces forward (dodawanie szumu)
- Proces reverse (przewidywanie i usuwanie szumu)
- Iteracyjne próbkowanie

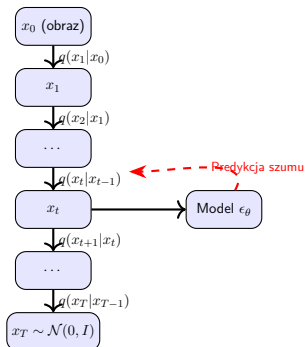
**Zalety:** wysoka jakość generowanych próbek, stabilność treningu

**Wady:** długi czas generowania, złożoność obliczeniowa

**Bibliografia:** [3]



## Diffusion Model - Funkcje straty (1/2)



### Struktura modelu dyfuzji: Model

dyfuzji opiera się na procesie forward, który stopniowo dodaje szum do danych, oraz na modelu  $\epsilon_\theta$ , który przewiduje szum w procesie reverse.

Proces obejmuje wiele kroków, od czystego obrazu  $x_0$  do czystego szumu  $x_T$ .

## Diffusion Model - Funkcje straty (2/2)

**Proces forward (dodawanie szumu):**

$$\begin{aligned}q(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t\mathbf{I}) \\q(x_t|x_0) &= \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I})\end{aligned}$$

**Parametryzacja:**

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

gdzie  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

**Funkcja straty:**

$$\mathcal{L}_{simple} = \mathbb{E}_{t, x_0, \epsilon} \left[ \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

**Parametry:**

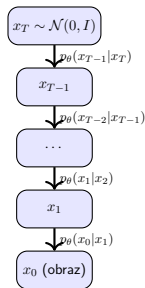
- $\beta_t$  - harmonogram szumu
- $\bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s)$
- $\epsilon_\theta$  - model predykcji szumu

## Diffusion Model - Próbkowanie

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

### Algorytm próbkowania:

- 1  $x_T \sim \mathcal{N}(0, \mathbf{I})$
- 2 dla  $t = T, T-1, \dots, 1$ :
  - Oblicz  $\mu_{\theta}(x_t, t)$
  - $z \sim \mathcal{N}(0, \mathbf{I})$  jeśli  $t > 1$ ,  
inaczej  $z = 0$
  - $x_{t-1} = \mu_{\theta}(x_t, t) + \sigma_t z$
- 3 Zwróć  $x_0$



### Kluczowe równanie:

$$\mu_{\theta}(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(x_t, t) \right)$$

gdzie  $\alpha_t = 1 - \beta_t$

## Porównanie modeli generatywnych - tabela

Model	Zalety	Wady
Autoencoder (2006)	Prostota implementacji, szybki trening	Słaba generatywność, rozmyte obrazy
VAE (2013)	Solidne podstawy teoretyczne, ciągła przestrzeń latentna	Rozmyte obrazy, trudność balansowania rekonstrukcji i KL divergencji
GAN (2014)	Ostre, realistyczne próbki	Niestabilność treningu, mode collapse
Conditional VAE (2015)	Kontrola nad procesem generacji, warunkowanie na klasach	Większa złożoność implementacji, wymaga etykiet
VQ-VAE (2017)	Ostrzejsze obrazy, dobra kompresja	Trudniejszy do trenowania, problemy z kwantyzacją
Diffusion (2020)	Najlepsza jakość obrazów, stabilny trening	Powolne próbkowanie, wysoka złożoność obliczeniowa

## Wyzwania implementacyjne

- **Dobór architektury:** Liczba warstw, liczba neuronów, funkcje aktywacji
- **Dobór wymiarowości przestrzeni latentnej:** Zbyt mała - utrata informacji, zbyt duża - brak generalizacji
- **Balansowanie funkcji straty:** Np. w VAE balans między rekonstrukcją a regularyzacją KL

$$\mathcal{L}_{VAE}(\beta) = \|x - \hat{x}\|^2 + \beta \cdot D_{KL}$$

- **Stabilność treningu:** Szczególnie w przypadku GAN-ów

$$\mathcal{L}_D = -\mathbb{E}_x[\log D(x)] - \mathbb{E}_z[\log(1 - D(G(z)))]$$

$$\mathcal{L}_G = -\mathbb{E}_z[\log D(G(z))]$$

- **Efektywność obliczeniowa:** Modele dyfuzji wymagają wielu kroków podczas generowania
- **Ocena jakości wygenerowanych próbek:** Metody ilościowe vs jakościowe



## Kompromisy w modelowaniu

### Rekonstrukcja vs różnorodność:

- W VAE:  $\beta$ -VAE

$$\mathcal{L}_{\beta\text{-VAE}} = \|x - \hat{x}\|^2 + \beta \cdot D_{KL}$$

- $\beta < 1$ : lepsza rekonstrukcja
- $\beta > 1$ : większa regularyzacja, lepsze generowanie

### Rate-distortion trade-off:

$$\mathcal{L} = \text{Rate} + \beta \cdot \text{Distortion}$$

### Wymiar przestrzeni latentnej:

- Mały wymiar  $\rightarrow$  silna kompresja, ale utrata szczegółów
- Duży wymiar  $\rightarrow$  lepsza rekonstrukcja, ale słabsza generalizacja

### Harmonogram szumu w Diffusion:

$$\beta_1, \beta_2, \dots, \beta_T$$

- Liniowy vs nieliniowy
- Wpływ na jakość generowania

# Przykłady zastosowań

## Generowanie danych syntetycznych:

- Augmentacja danych w uczeniu maszynowym

$$\mathcal{D}_{aug} = \mathcal{D} \cup \{G(z_i) | z_i \sim p(z)\}_{i=1}^N$$

- Syntetyczne dane dla trenowania innych modeli
- Generowanie przykładów do zastosowań edukacyjnych

## Matematycznie:

$$p_{model}(x) \approx p_{data}(x)$$

## Zastosowania praktyczne:

- Transfer stylu pisma

$$\hat{x} = g(f(x_{style}), y_{content})$$

- Uzupełnianie brakujących fragmentów

$$\hat{x}_{full} = \arg \max_x p(x | x_{observed})$$

- Korekta i poprawa pisma odręcznego
- Konwersja cyfr między różnymi stylami

$$x_{target} = g(f(x_{source}), y_{target})$$

## Bibliografia I

- [1] Ian Goodfellow i in. “Generative adversarial nets”. W: *Advances in neural information processing systems*. 2014, s. 2672–2680.
- [2] Geoffrey E Hinton i Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. W: *Science* 313.5786 (2006), s. 504–507.
- [3] Jonathan Ho, Ajay Jain i Pieter Abbeel. “Denoising diffusion probabilistic models”. W: *Advances in neural information processing systems*. T. 33. 2020, s. 6840–6851.
- [4] Diederik P Kingma i Max Welling. “Auto-encoding variational bayes”. W: *arXiv preprint arXiv:1312.6114* (2013).
- [5] Aaron van den Oord, Oriol Vinyals i Koray Kavukcuoglu. “Neural discrete representation learning”. W: *Advances in neural information processing systems*. 2017, s. 6306–6315.

## Bibliografia II

- [6] Kihyuk Sohn, Xinchun Yan i Honglak Lee. “Learning structured output representation using deep conditional generative models”. *W: Advances in neural information processing systems*. 2015, s. 3483–3491.