Uczenie Maszynowe 2

Modele generatywne

dr hab. Piotr Duda, prof. AGH i PCz

Plan wykładu

Wykład 1

- **Organiczone Maszyny Boltzmanna**

Wykład 2

- **Autoenkodery, VAE, WAE**

Wykład 3

- **GAN, Normalizing Flows**

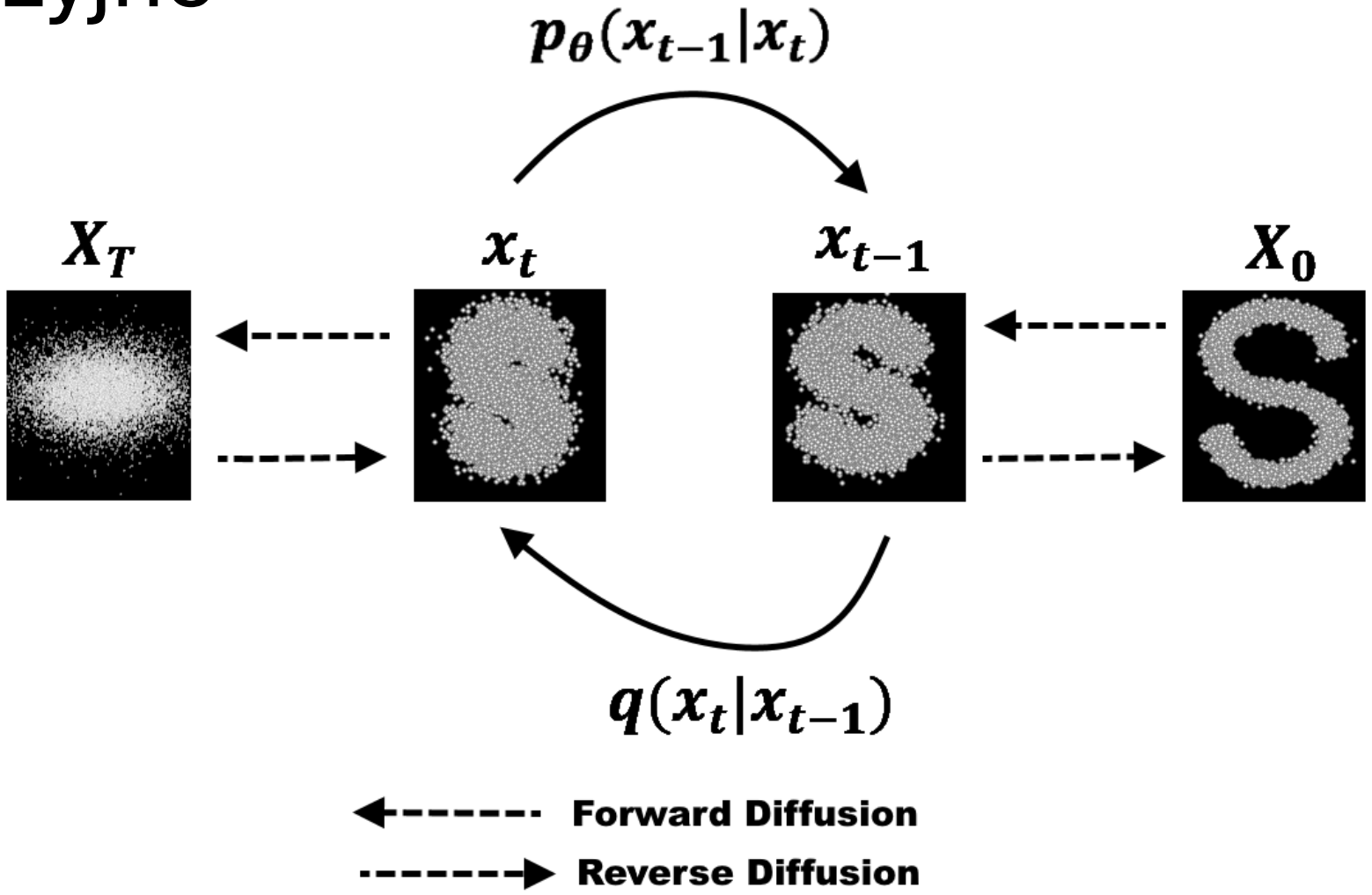
Wykład 4

- **Podejście dyfuzyjne, Przegląd „aktualnych” modeli**

Wykład 4 – Modele generatywne

- Podejście dyfuzyjne
- Transformery
- Podsumowanie metod
- Przegląd „aktualnych” rozwiązań
- Kolejny krok…

Modele dyfuzyjne



Źródło: https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da

Koder

Pierwsze przekształcenie:

$$z_1 = \sqrt{1-\beta_1}x + \sqrt{\beta_1}\epsilon_1$$

Kolejne przekształcenia:

$$z_t = \sqrt{1-\beta_t}z_{t-1} + \sqrt{\beta_t}\epsilon_t,$$

dla t=2,3,…,T, $\beta_t \in [0,1]$, $\epsilon_t \sim N(0,1)$.

Koder

$$P(z_1|x), P(z_2|z_1), \dots, P(z_T|z_{T-1})$$

$$P(z_1, \dots, z_T|x) = P(z_1|x) \prod_{i=2}^{T} P(z_i|z_{i-1})$$

Diffusion kernel

$$z_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon,$$

gdzie

$$\alpha_t = \prod_{i=1}^{t} 1 - \beta_i \,,$$

a $\epsilon \sim N(0,1)$

Prawdopodobieństwo...

...wygenerowania bardziej zaszumionego obrazu

$$P(z_1|x) \Rightarrow N(\sqrt{1-\beta_1}x, \beta_1 I)$$

$$P(z_t|z_{t-1}) \Rightarrow N(\sqrt{1-\beta_t}z_{t-1}, \beta_t I)$$

... wygenerowania mniej zaszumionego obrazu

$$P(z_{t-1}|z_t) = \frac{P(z_t|z_{t-1})P(z_{t-1})}{P(z_t)}$$

Dekoder

$$P(z_T) \Rightarrow N(0, I)$$

$$P(z_{t-1}|z_t, \varphi_t) \Rightarrow N\left(f_t(z_t, \varphi_t), \sigma_t^2 I\right)$$

$$P(x|z_1, \varphi_1) \Rightarrow N(f_1(z_1, \varphi_1), \sigma_1^2 I)$$

Uczenie modelu

$$\underset{\varphi_t}{argmax} \sum_{i=1}^{n} \log P(x_i|\varphi_t)$$

Załóżmy, że $z_{1..T}$ to wielowymiarowa zmienna losowa pochodząca z rozkładu łącznego zmiennych $z_1, \ldots, z_T$. Wówczas

$$\int P(z_{1..T}|x) \log\left(\frac{P(x, z_{1..T}|\varphi_{1,\ldots,T})}{P(z_{1..T}|x)}\right) dz_{1..T} \leq \log P(x|\varphi_{1,\ldots,T})$$

Uczenie modelu

$$\int P(z_{1..T}|x) \log\left(\frac{P(x, z_{1..T}|\varphi_{1,\dots,T})}{P(z_{1..T}|x)}\right) dz_{1..T} \approx$$

$$\mathbb{E}_{P(z_1|x)} \log\big(P(x|z_1, \varphi_1)\big) - \sum_{i=2}^{T} \mathbb{E}_{P(z_i|x)} D_{KL}(P(z_{i-1}|z_i, x)||P(z_{i-1}|z_i, \varphi_i))$$

Uczenie modelu

Diffusion loss function

$$L[\boldsymbol{\phi}_{1\ldots T}] = \sum_{i=1}^{I}\Bigg(\overbrace{-\log\Big[\mathrm{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big]}^{\text{reconstruction term}} + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\Big\|\underbrace{\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_{it}+\frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_i}_{\text{target, mean of } q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})} - \underbrace{\mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]}_{\text{predicted } \mathbf{z}_{t-1}}\Big\|^2\Bigg),$$

Modyfikacja diffusion loss

Pamiętając, że

$$z_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon,$$

mamy

$$x = \frac{1}{\sqrt{\alpha_t}} z_t + \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\epsilon$$

Uczenie modelu

Diffusion loss function

$$
\begin{aligned}
L[\boldsymbol{\phi}_{1\ldots T}] = \sum_{i=1}^{I}\Bigg(&-\log\Big[\mathrm{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] \\
&+\sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_{it}-\frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}_{it}\right)-\mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]\right\|^2\Bigg).
\end{aligned}
$$

Pamiętając, że

$$z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\epsilon_t$$

mamy

$$z_{t-1} = \frac{1}{\sqrt{1-\beta_t}} z_t + \frac{\sqrt{\beta_t}}{\sqrt{1-\beta_t}} \epsilon_t$$

Pamiętając, że

$$z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\epsilon_t$$

mamy

$$z_{t-1} = \frac{1}{\sqrt{1-\beta_t}} z_t + \frac{\sqrt{\beta_t}}{\sqrt{1-\beta_t}} \epsilon_t$$

A skoro $z_{t-1} = f_t(z_t, \varphi_t)$ to

$$z_{t-1} = \frac{1}{\sqrt{1-\beta_t}} z_t + \frac{\sqrt{\beta_t}}{\sqrt{1-\beta_t}} g_t(z_t, \varphi_t)$$

Uczenie modelu

Diffusion loss function

$$
\begin{aligned}
L[\boldsymbol{\phi}_{1\ldots T}] &= \sum_{i=1}^{I}\sum_{t=1}^{T}\left\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]-\boldsymbol{\epsilon}_{it}\right\|^2 \\
&= \sum_{i=1}^{I}\sum_{t=1}^{T}\left\|\mathbf{g}_t\left[\sqrt{\alpha_t}\cdot\mathbf{x}_i+\sqrt{1-\alpha_t}\cdot\boldsymbol{\epsilon}_{it},\boldsymbol{\phi}_t\right]-\boldsymbol{\epsilon}_{it}\right\|^2,
\end{aligned}
$$

Uczenie modelu

Algorithm 18.1: Diffusion model training

Input: Training data $\mathbf{x}$
Output: Model parameters $\boldsymbol{\phi}_t$
repeat
  for $i \in \mathcal{B}$ **do** // For every training example index in batch
    $t \sim \text{Uniform}[1, \ldots T]$ // Sample random timestep
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ // Sample noise
    $\ell_i = \left\| \mathbf{g}_t\left[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\phi}_t\right] - \boldsymbol{\epsilon} \right\|^2$ // Compute individual loss
  Accumulate losses for batch and take gradient step
until converged

Generowanie danych

Algorithm 18.2: Sampling

Input: Model, $\mathbf{g}_t[\bullet, \boldsymbol{\phi}_t]$

Output: Sample, $\mathbf{x}$

$\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ // Sample last latent variable

for $t = T \ldots 2$ **do**

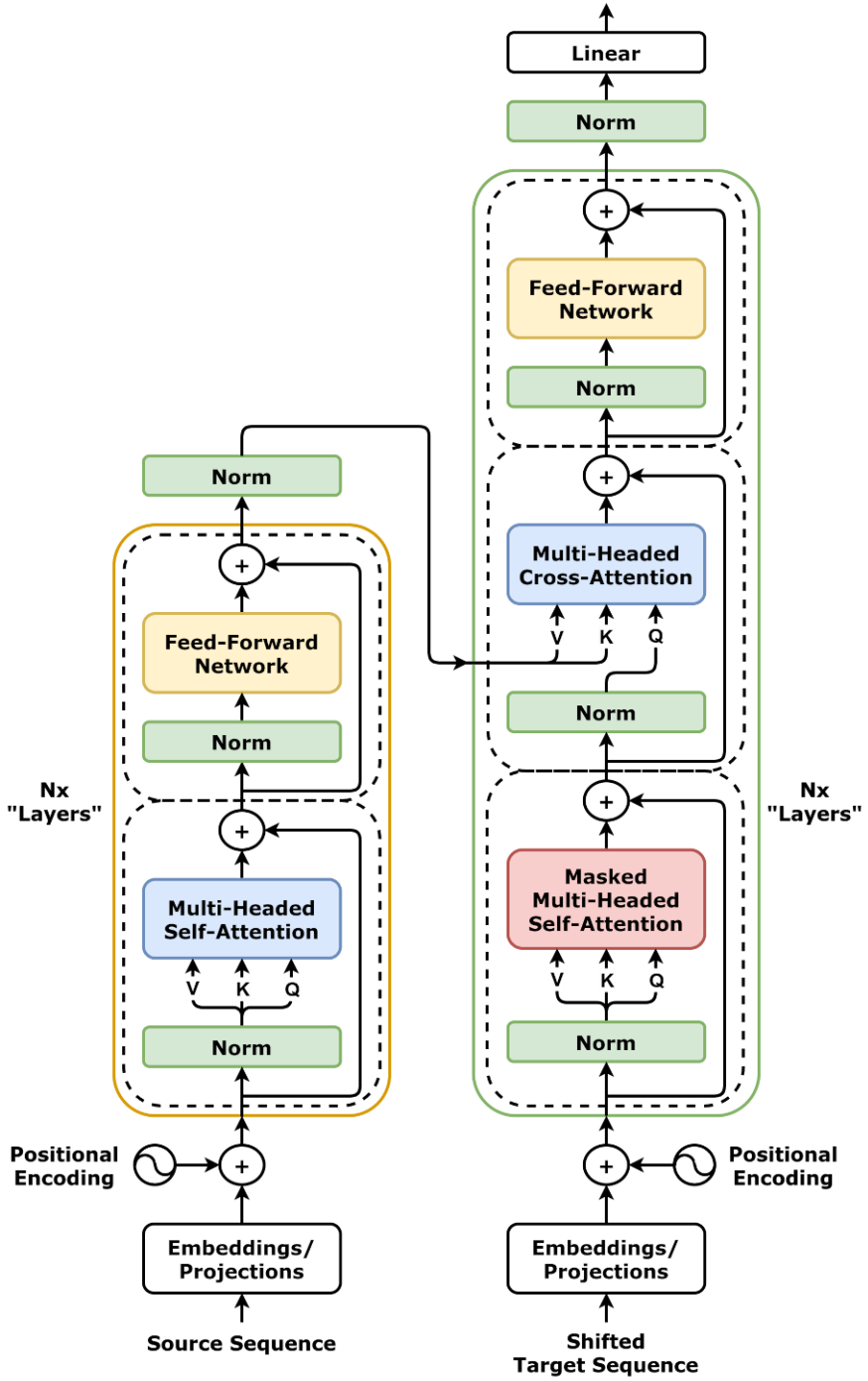
$\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ // Predict previous latent variable

$\boldsymbol{\epsilon} \sim \text{Norm}_{\boldsymbol{\epsilon}}[\mathbf{0}, \mathbf{I}]$ // Draw new noise vector

$\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t\boldsymbol{\epsilon}$ // Add noise to previous latent variable

$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_1, \boldsymbol{\phi}_1]$ // Generate sample from $\mathbf{z}_1$ without noise

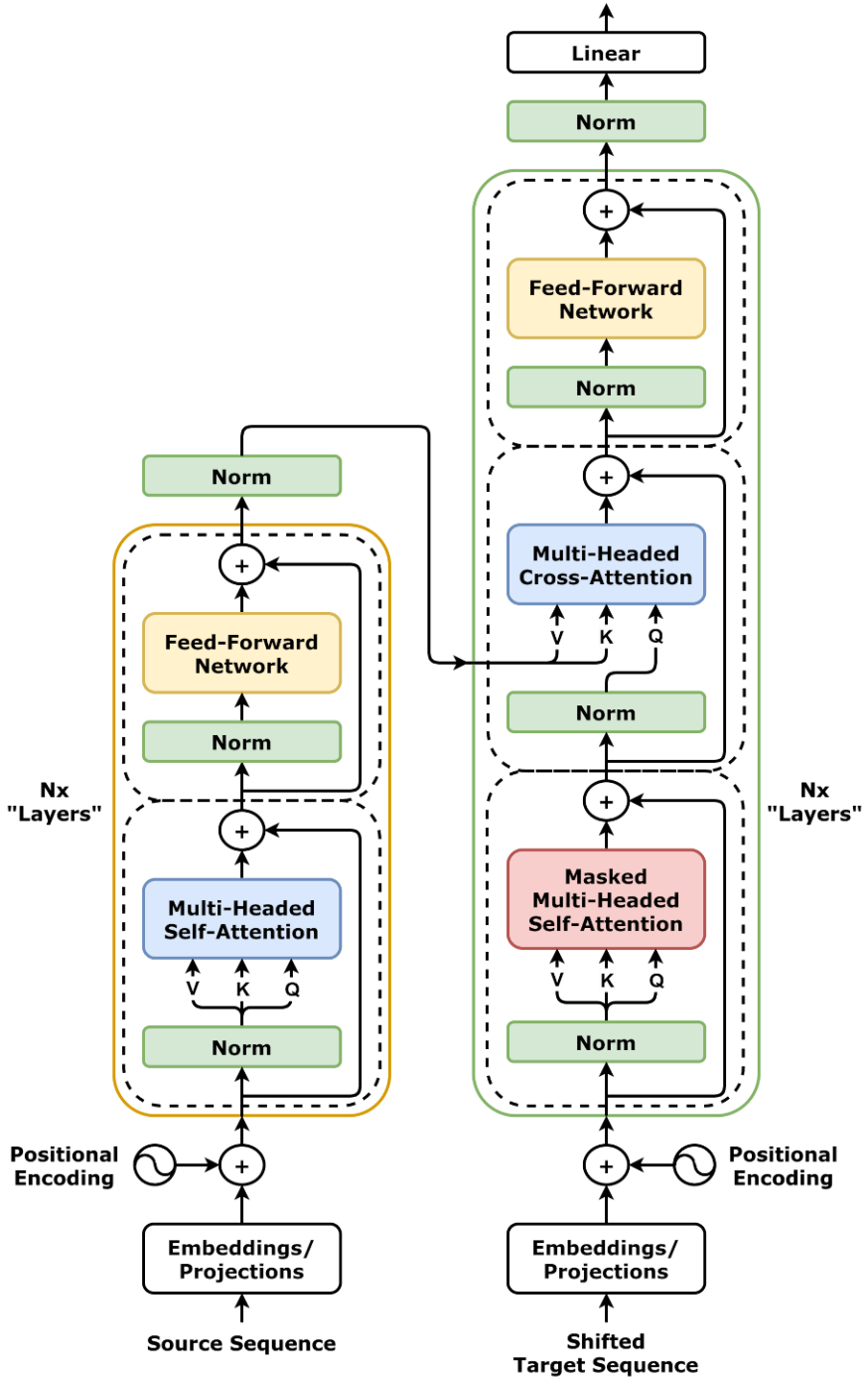
Transformery



Transformery

Struktura transformera

Encoder-Decoder

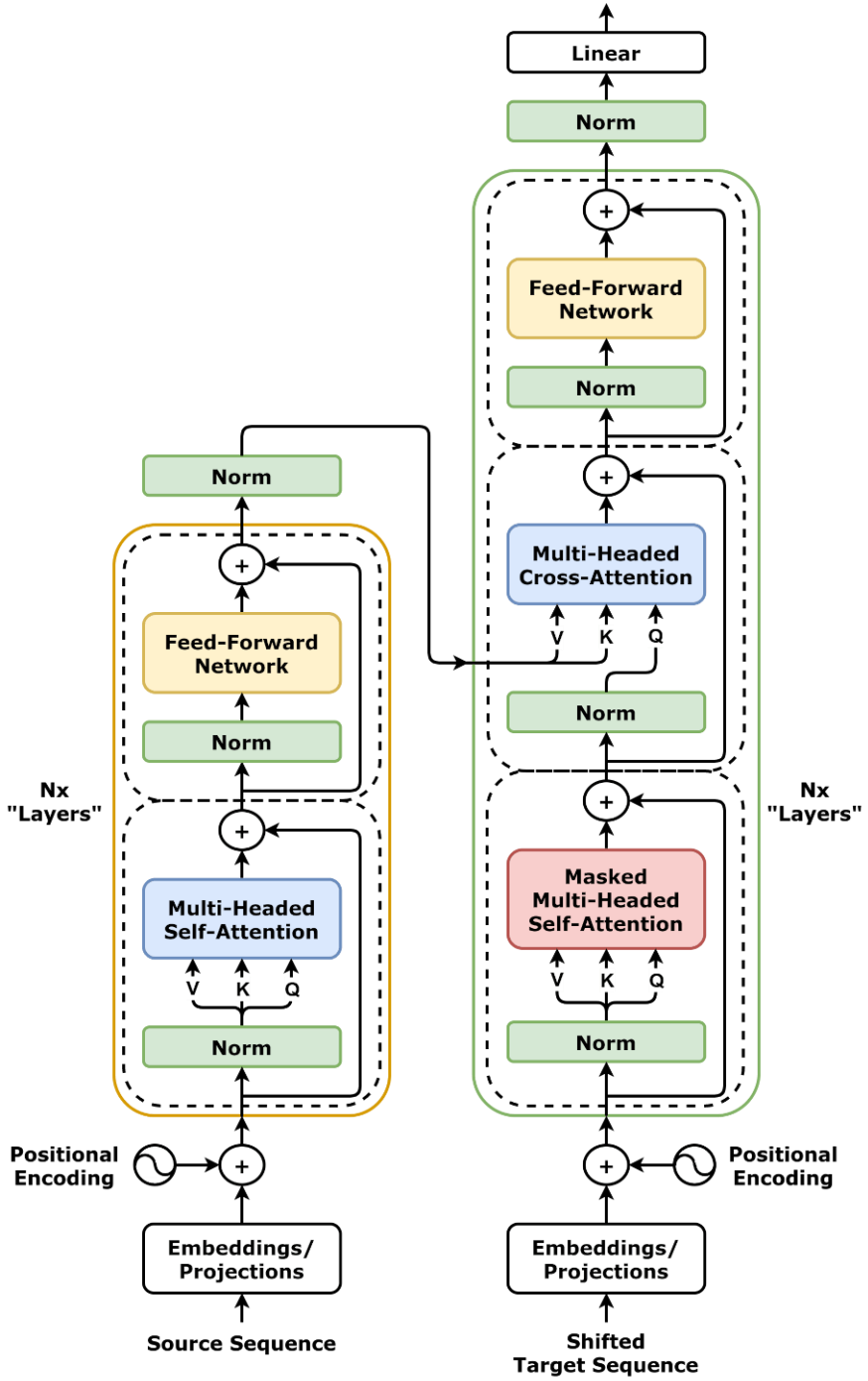


Transformery

Struktura transformera

Encoder-Decoder

Elementy struktur



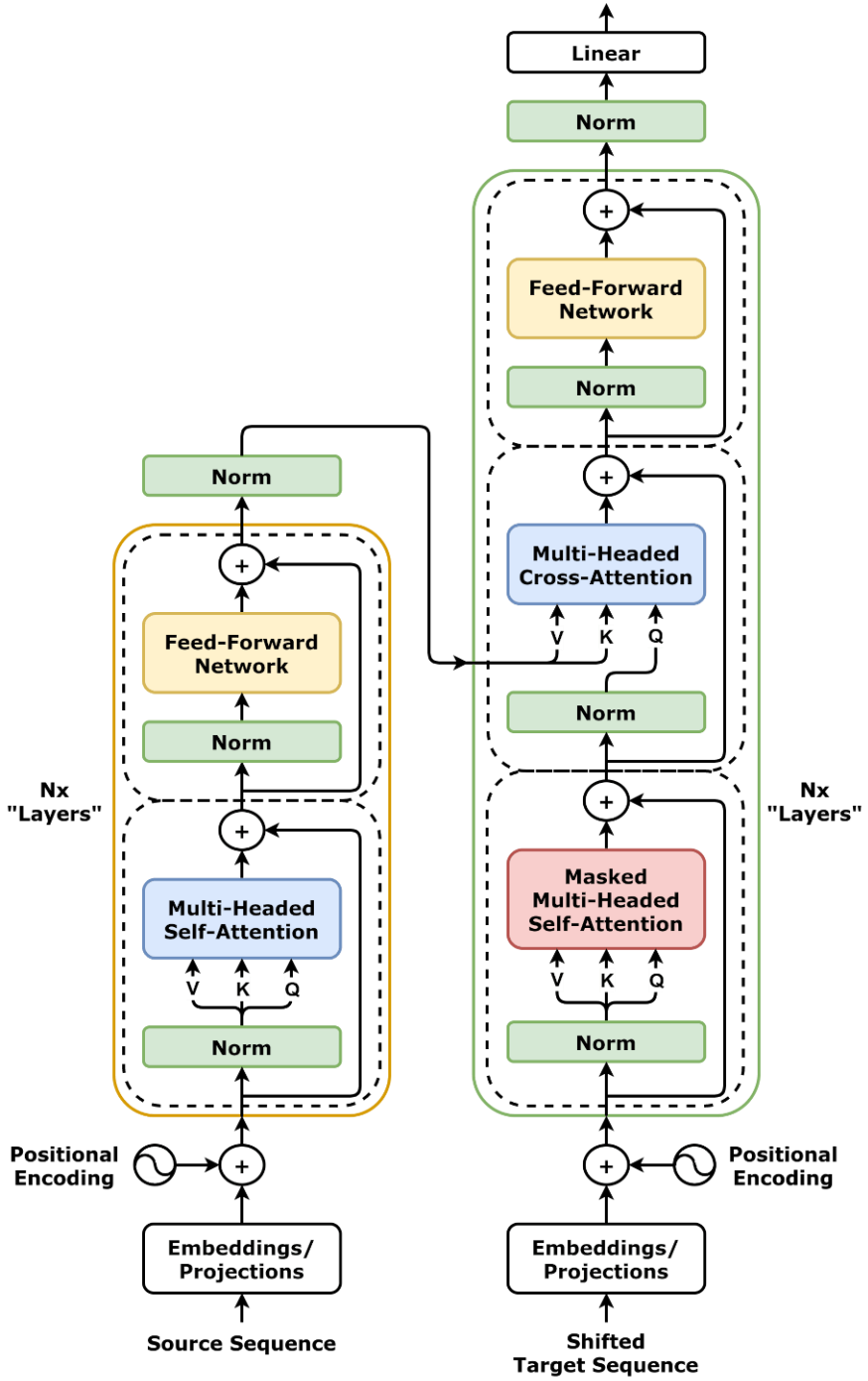
Transformery

Struktura transformera

Encoder-Decoder

Elementy struktur:

- Multi-head attention



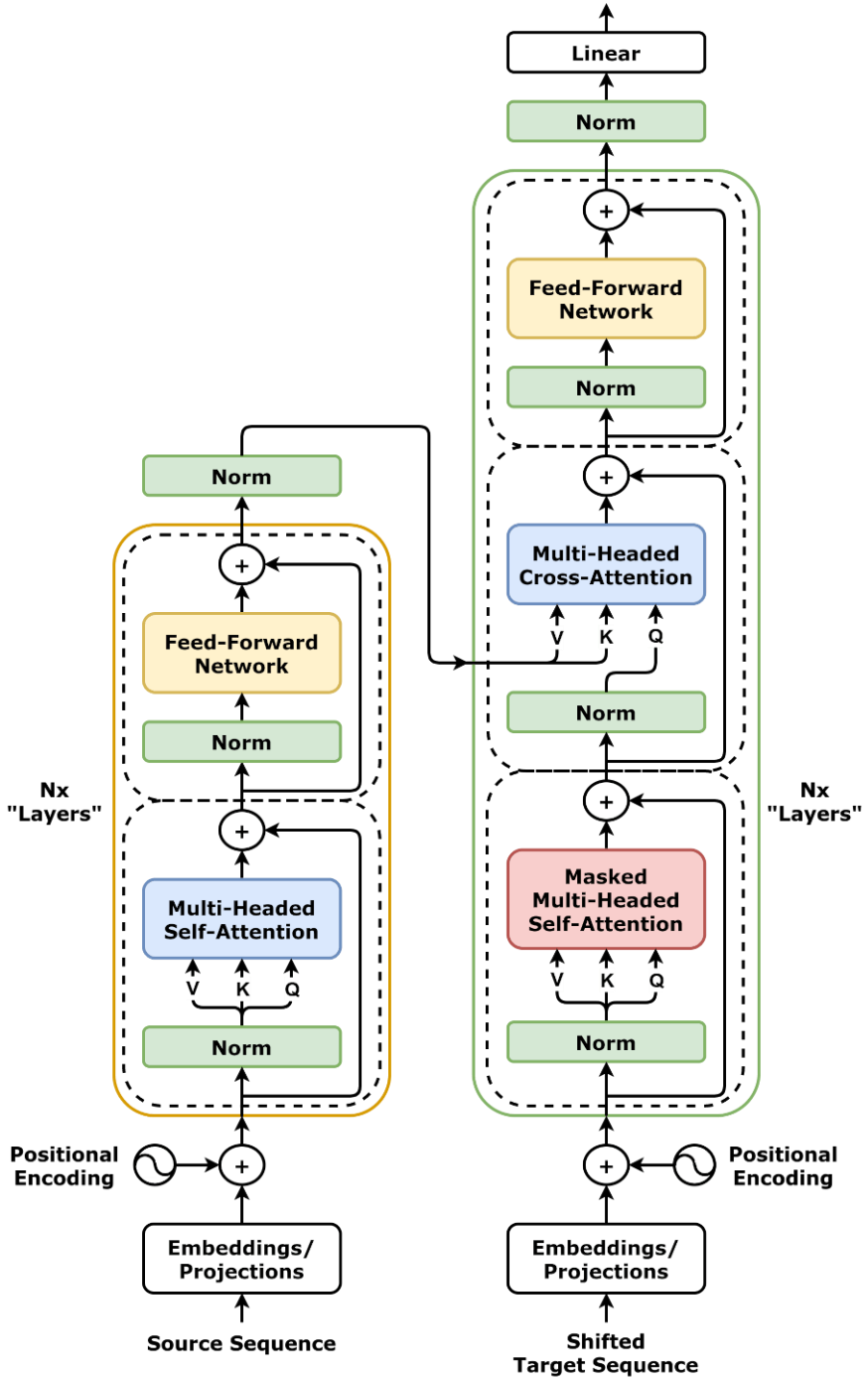
Transformery

Struktura transformera

Encoder-Decoder

Elementy struktur:

- Multi-head attention
- Embeddingi



Transformery

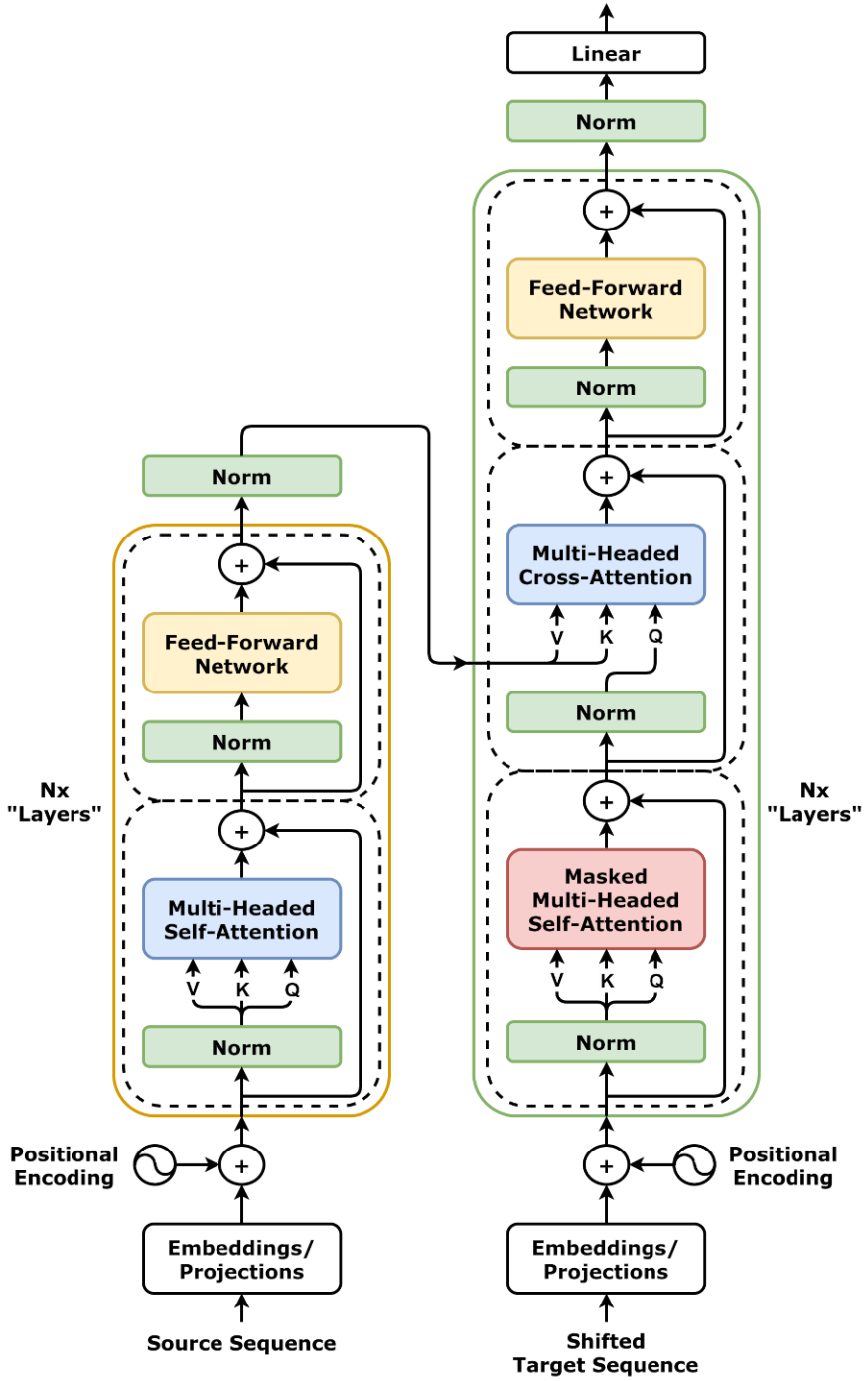
Struktura transformera

Encoder-Decoder

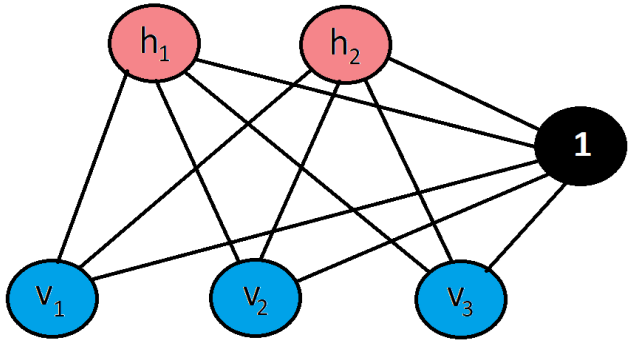
Elementy struktur:

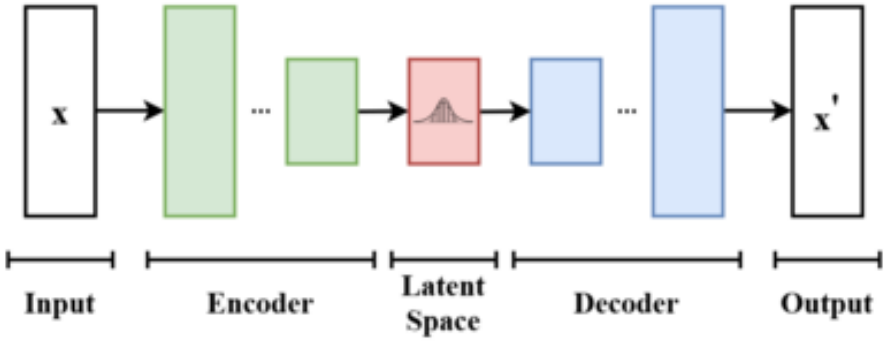
- Multi-head attention
- Embeddingi

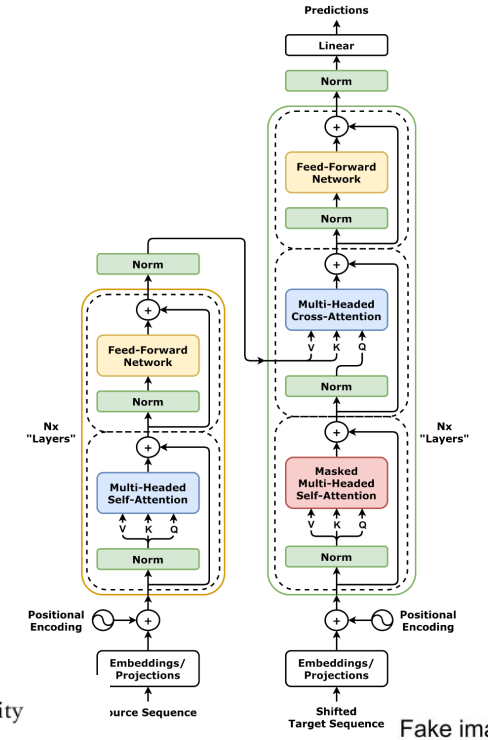
Uczenie

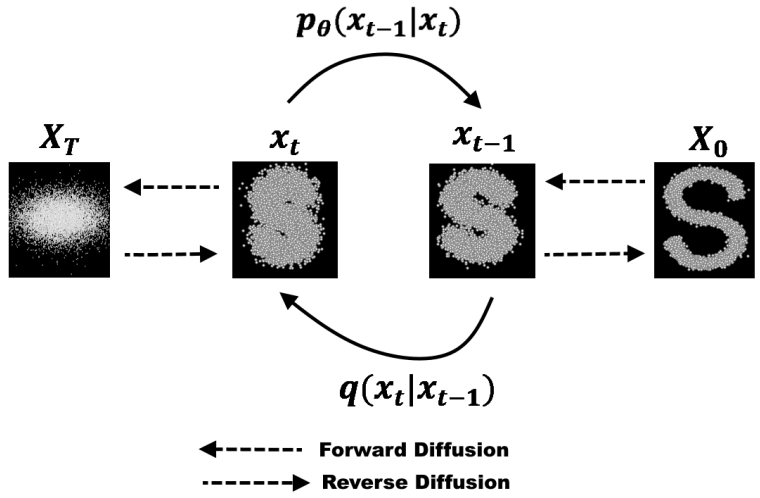


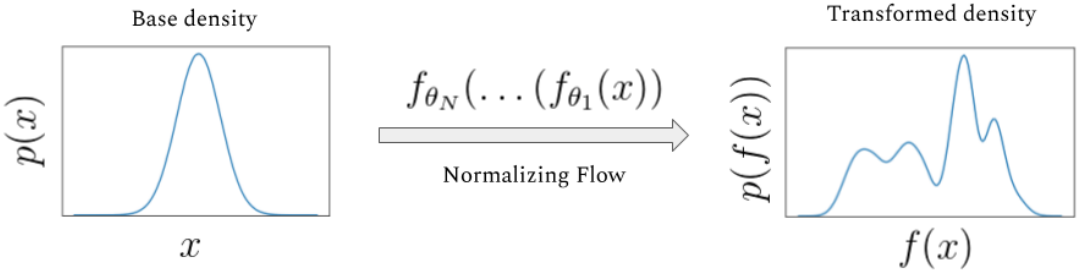
Podsumowanie

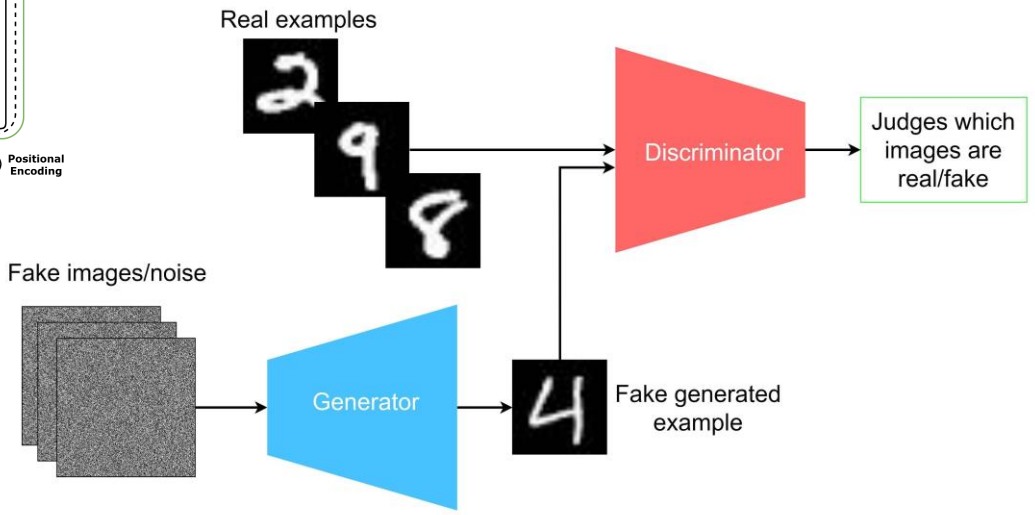












Porównanie podejść generacyjnych zalety - wady

Metoda	Zalety	Wady
Modele dyfuzyjne	- Wysoka jakość generowanych próbek - Stabilność treningu - Umożliwiają iteracyjną kontrolę nad generacją	- Wysokie wymagania obliczeniowe - Powolny proces generacji (wiele kroków samplingu)
Normalizing Flows	- Łatwość obliczenia prawdopodobieństwa danych - Odwracalność modelu	- Trudność w skalowaniu do bardziej złożonych danych - Wymagają bardzo dokładnych parametrów
GAN	- Szybka generacja danych - Wysoka jakość próbek, zwłaszcza obrazów	- Trudność w stabilnym trenowaniu (problem zbalansowania generatora i dyskryminatora) - Mniej intuicyjna kontrola
VAE	- Łatwość implementacji - Naturalne modelowanie zmiennych latentnych	- Generowane próbki mogą być mniej realistyczne niż GAN - Rozmyte wyniki dla obrazów
WAE	- Lepsza jakość niż klasyczne VAE - Stabilniejszy trening	- Wciąż słabsza jakość generacji niż GAN - Problemy z kosztami obliczeń dla dużych zbiorów danych
RBM	- Prosta implementacja dla małych zbiorów danych - Dobre do modelowania nieliniowych relacji	- Problemy ze skalowalnością - Słaba wydajność przy bardziej złożonych danych

Porównanie podejść generacyjnych
Zastosowania

Modele dyfuzyjne: Nadają się do aplikacji wymagających wysokiej jakości i elastyczności generacji, np. edycji obrazów.

Normalizing Flows: Używane, gdy obliczenie dokładnego prawdopodobieństwa danych jest istotne (np. w analizie anomalii).

GAN: Idealne do generowania realistycznych obrazów. Popularne w grafice komputerowej i sztuce cyfrowej.

VAE: Sprawdzają się w analizie danych i uczeniu reprezentacji, gdzie zmienne latentne są ważne.

WAE: Dobry wybór dla kompromisu między prostotą a jakością generowanych danych.

RBM: Raczej przestarzałe, dobre dla małych zbiorów danych lub nauki podstawowych zasad generatywnych.

Porównanie podejść generacyjnych
Implementacje

Metoda	implementacja	Krótki opis
Modele dyfuzyjne	Stable Diffusion / DALL-E 2	Modele do generacji obrazów wysokiej jakości z opisów tekstowych.
Normalizing Flows	RealNVP / Glow	RealNVP wprowadził odwracalne transformacje, a Glow poprawił ich efektywność.
GAN	StyleGAN / BigGAN	StyleGAN jest używany do realistycznej generacji obrazów twarzy, BigGAN do dużych zbiorów danych.
VAE	Beta-VAE	Modyfikacja klasycznego VAE z regulacją wpływu zmiennej latentnej na generację.
WAE	WAE-GAN / WAE-MMD	Połączenie idei autoenkoderów z dystansem Wassersteina dla lepszej jakości wyników.
RBM	Deep Belief Networks (DBN)	Rozszerzenie RBM do głębszych struktur, używane w analizie danych.

Obszary zastosowań a modele

Zastosowanie	Model / Metoda
Generowanie obrazów	Stable Diffusion
	StyleGAN
Generowanie tekstu	GPT
	VAE-LSTM
Generowanie dźwięku / mowy	WaveNet
	DiffWave
Generowanie muzyki	MuseNet
	Jukebox
Obróbka i edycja obrazów	DALL-E
	CycleGAN

Obszary zastosowań a modele

Zastosowanie	Model / Metoda
Uczenie reprezentacji danych	Beta-VAE
	Deep Belief Networks
Wykrywanie anomalii	Glow
	VAE
Tworzenie 3D	NeRF
	GAN3D
Generowanie wideo	Video Diffusion Models
	MoCoGAN
Symulacje w nauce	Schrodinger Bridge Models
	PhysicsGAN

Przegląd implementacji

Stable Diffusion 3.5

https://stability.ai/news/introducing-stable-diffusion-3-5

StyleGAN

https://github.com/NVlabs/stylegan

GPT

https://github.com/openai/gpt-2

WaweNet

https://deepmind.google/discover/blog/wavenet-a-generative-model-for-raw-audio/

Przegląd implementacji

MuseNet

https://openai.com/index/musenet/

Jukebox

https://openai.com/index/jukebox/

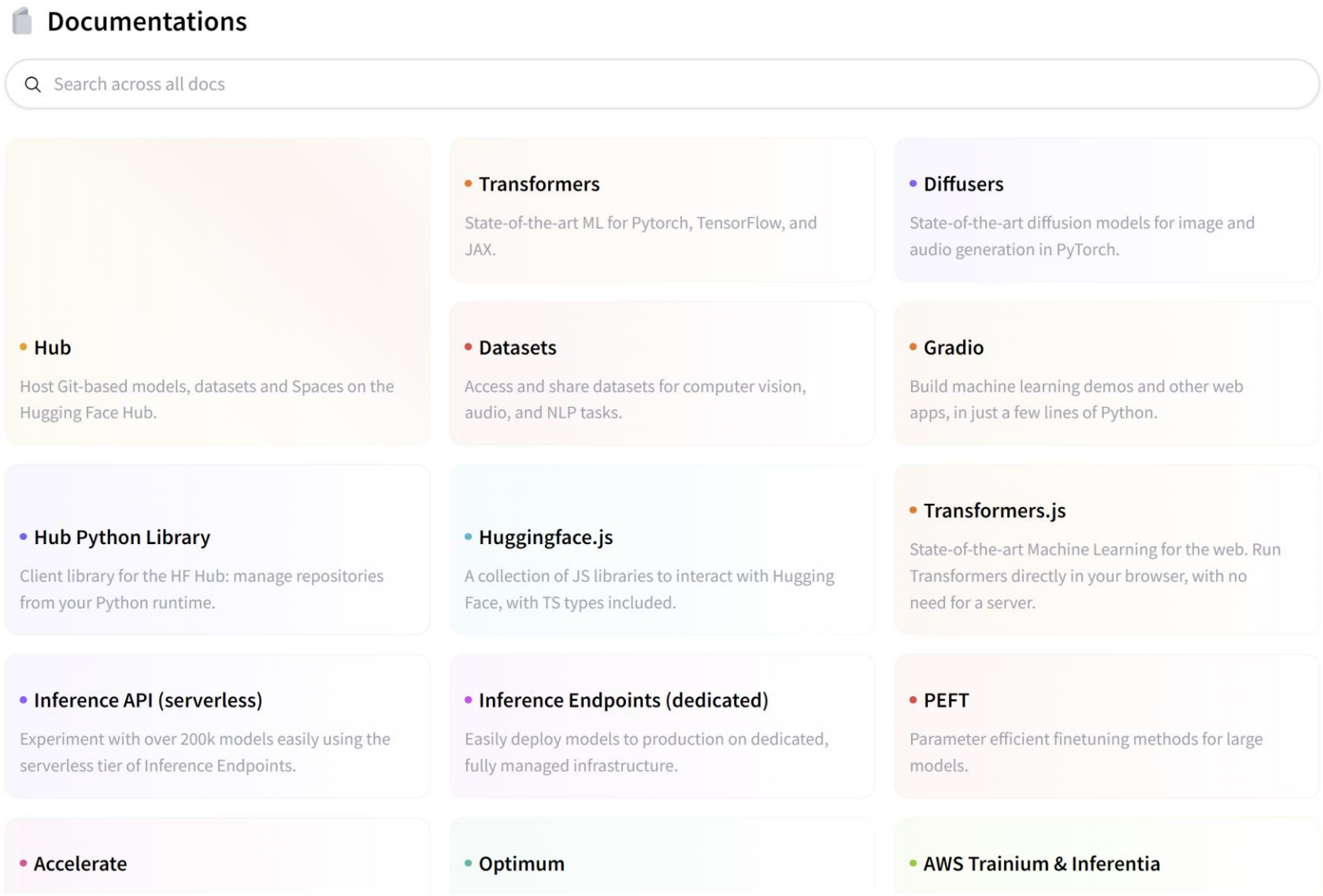
DALL-E

https://openai.com/index/dall-e-3/

CycleGAN

https://junyanz.github.io/CycleGAN/

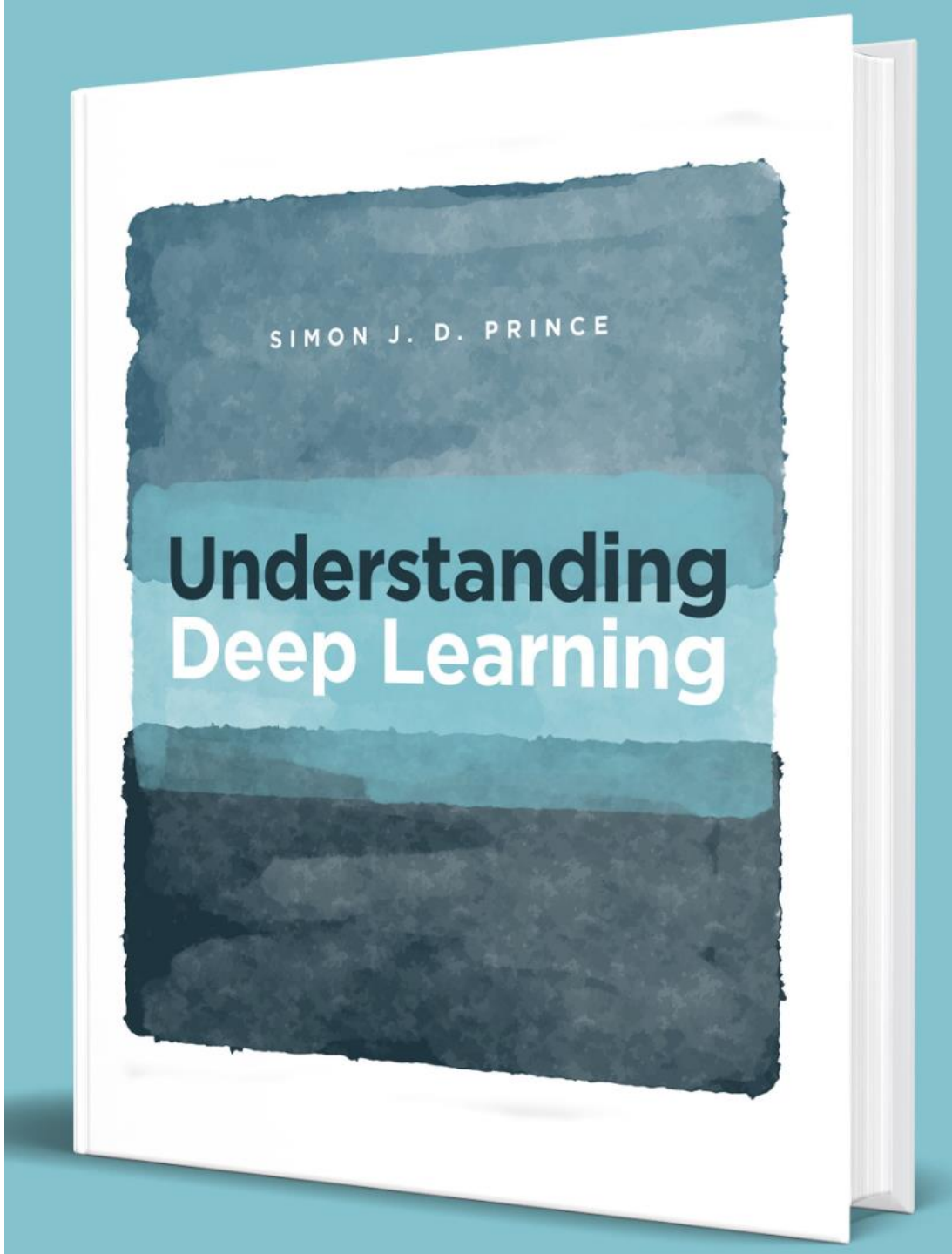
Od czego zacząć?



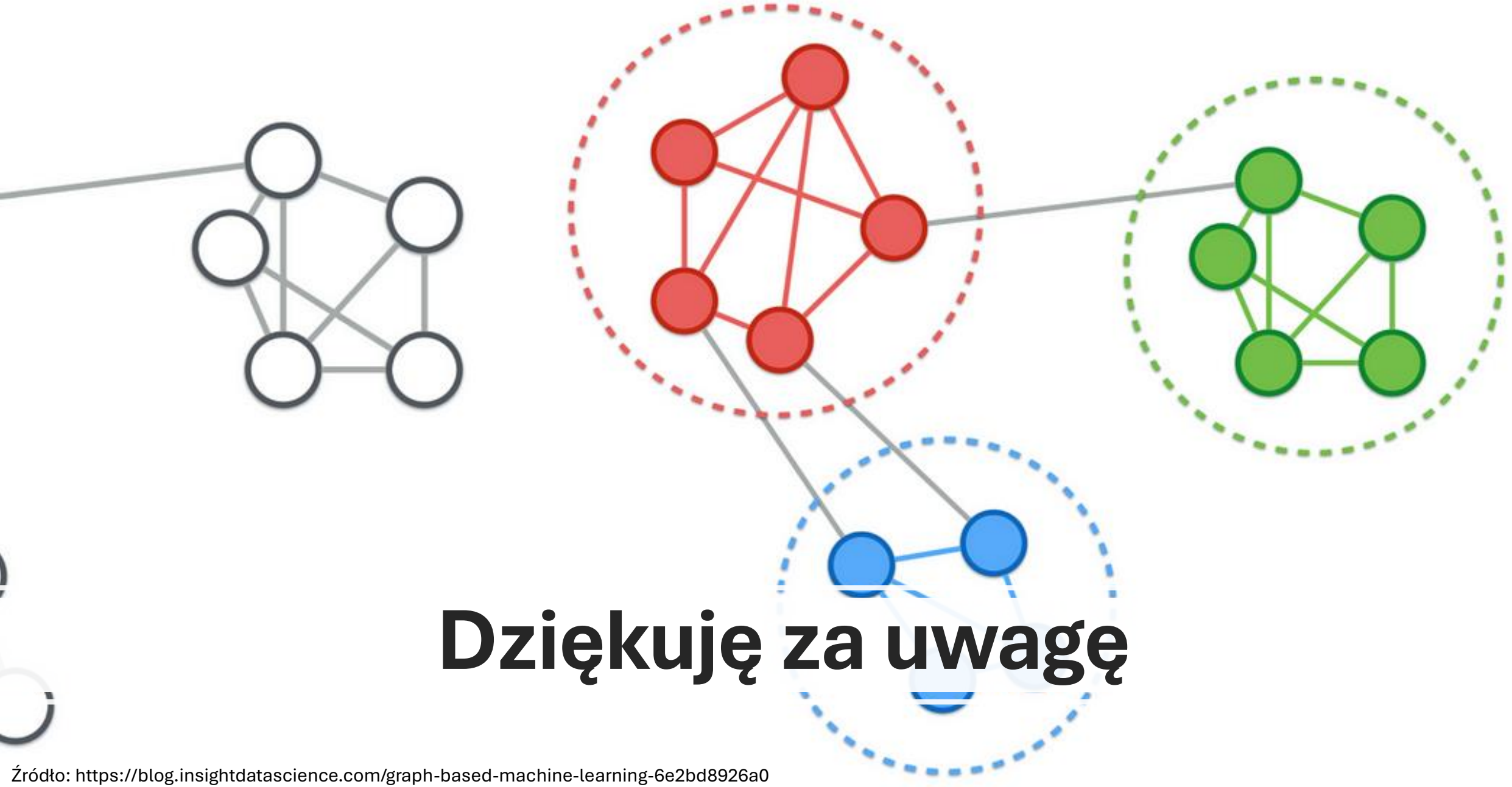
Gdzie szukać teorii?

Simon JD. Prince,
Understanding deep learning.
MIT press, 2023.

https://udlbook.github.io/udlbook/



Pytania / Komentarze / Uwagi



Dziękuję za uwagę

Źródło: https://blog.insightdatascience.com/graph-based-machine-learning-6e2bd8926a0