

Wprowadzenie do transformerów... czyli kilka słów o mechanizmie atencji

Kamil Książek

Wydział Matematyki i Informatyki
Uniwersytet Jagielloński
Prof. Stanisława Łojasiewicza 6
30-348 Kraków
kamil.ksiazek@uj.edu.pl



Centrum Wiarygodnej Sztucznej Inteligencji
Politechnika Warszawska
Rektorska 4
00-614 Warszawa
kamil.ksiazek@pw.edu.pl



Centre for Credible
Artificial Intelligence

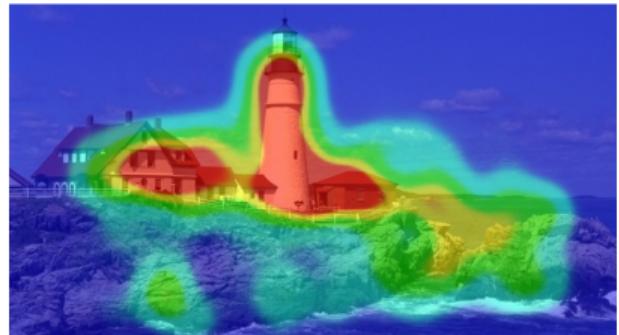
24 stycznia 2026

Agenda

- ▶ Wprowadzenie
- ▶ Definicja atencji
- ▶ Atencja wielogłowicowa
- ▶ Warstwa normalizacyjna
- ▶ Wektory pozycyjne
- ▶ Transformer wizyjny

Mechanizm atencji

- ▶ Mechanizm atencji (ang. *attention*) zaczął być stosowany w wizji komputerowej w 2014 roku, aby wskazać ważne regiony obrazu, które przyczyniają się do otrzymania pożądanego wyjścia. Ułatwia to użytkownikowi zrozumienie, gdzie i na czym skupiony jest model.
- ▶ W 2017 roku pojawiła się praca *Attention Is All You Need* autorstwa Ashisha Vaswaniego i in., która wprowadzała architekturę transformerów (ang. *transformers*) z użyciem mechanizmu atencji.



Źródło: P. Poupart, CS480/680 Lecture 19: Attention and Transformer Networks, Spring 2019, University of Waterloo.

Sieci rekurencyjne vs transformer

Wyzwania związane z sieciami rekurencyjnymi:

- ▶ długodystansowe zależności;
- ▶ eksplozja i zanik gradientów;
- ▶ duża liczba kroków treningowych;
- ▶ rekurencja uniemożliwia zrównoleglenie obliczeń.

Zalety transformerów:

- ▶ lepiej radzą sobie z długodystansowymi zależnościami;
- ▶ brak eksplozji i zaniku gradientów;
- ▶ mniejsza liczba kroków treningowych;
- ▶ brak rekurencji umożliwia zrównoleglenie obliczeń.

Definicja atencji

Mechanizm atencji naśladuje odzyskanie wartości v_i (ang. *value*) dla zapytania q (ang. *query*) w oparciu o klucz k_i (ang. *key*) z bazy danych.

Niech T będzie liczbą kluczy i odpowiadających im wartości w bazie, a $f(q, k_i)$ funkcją podobieństwa, gdzie $i \in \{1, \dots, T\}$. Wowczas

$$\text{Attention}(q, k, v) = f(q, k_1) \cdot v_1 + f(q, k_2) \cdot v_2 + \dots + f(q, k_T) \cdot v_T \quad (1)$$

Funkcje podobieństwa

Mierzymy podobieństwo między zapytaniem a poszczególnymi kluczami od 1 do T .

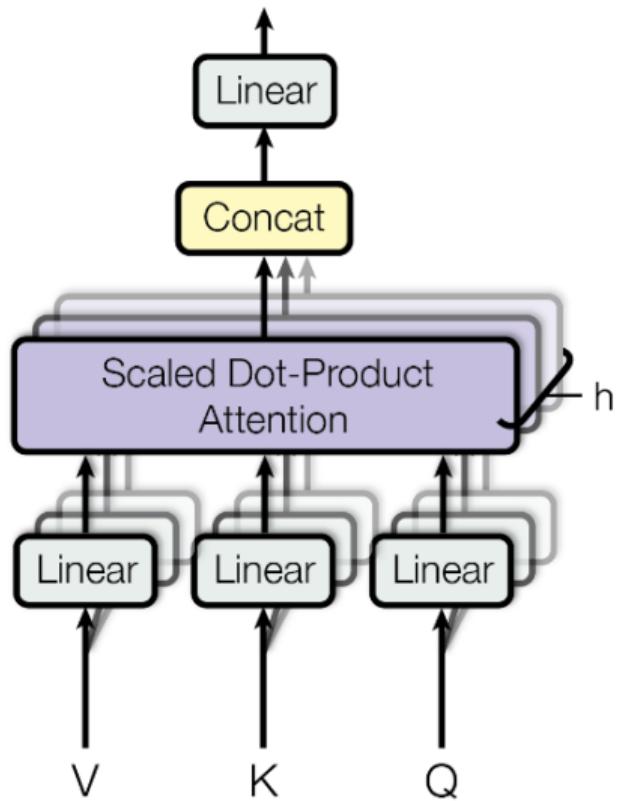
$$s_i = f(q, k_i) = \begin{cases} q^\top k_i & \text{iloczyn skalarny (ang. } dot \text{ product)} \\ \frac{q^\top k_i}{\sqrt{d}} & \text{przeskalowany iloczyn skalarny (ang. } scaled \text{ dot product)} \\ q^\top W k_i & \text{uogólniony iloczyn skalarny (ang. } general \text{ dot product)} \end{cases}$$

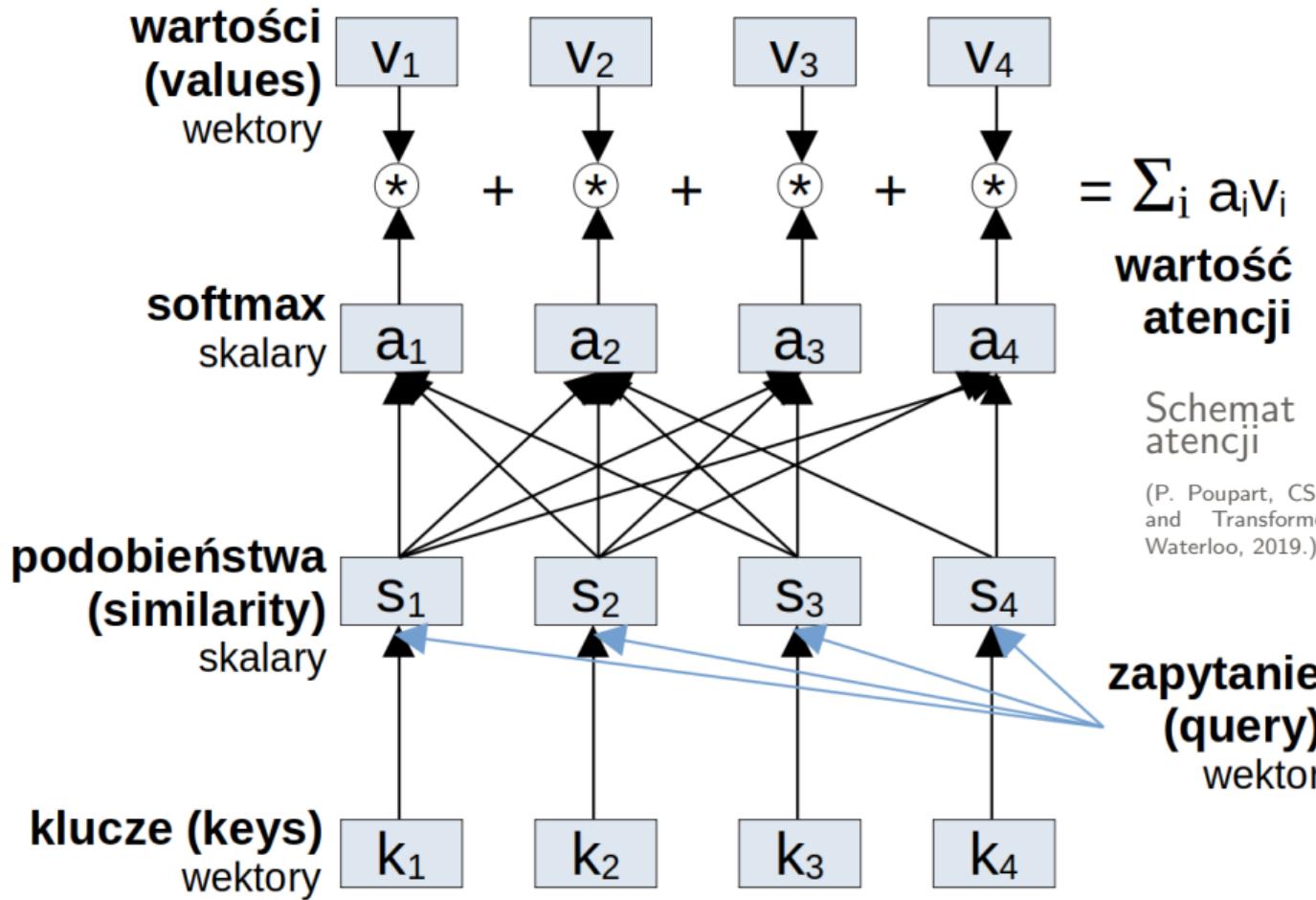
gdzie d jest wymiarowością klucza, zaś W jest macierzą wag optymalizowaną przez sieć.

Na policzone wartości funkcji podobieństwa nakładana jest funkcja softmax, tj.:

$$a_i = \frac{e^{s_i}}{e^{s_1} + e^{s_2} + \dots + e^{s_T}} \quad (2)$$

Mechanizm atencji

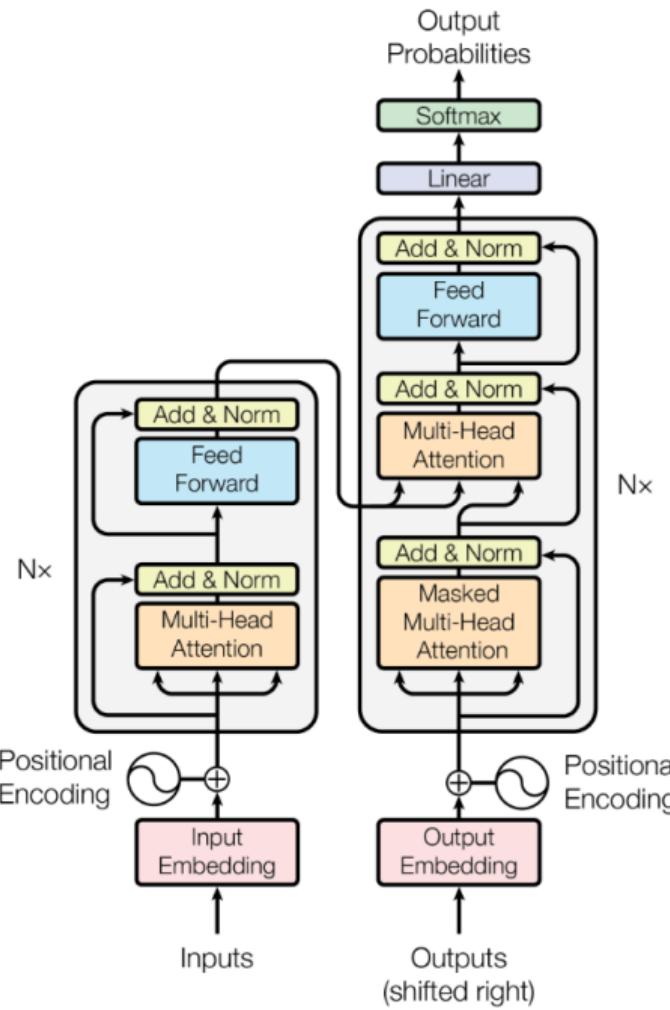




Schemat
atencji

mechanizmu

(P. Poupart, CS480/680 Lecture 19: Attention and Transformer Networks, University of Waterloo, 2019.)



Architektura transformera

(A. Vaswani et al, Attention Is All You Need, Proceedings of NeurIPS 2017.)

Multi-head attention

Atencja wielogłowicowa (ang. multi-head attention) sprowadza się do utworzenia wielu głowic atencji (dla zapytania) z różnymi wagami. W ten sposób łączymy informacje z różnych reprezentacji.

Niech h będzie liczbą głowic atencji, W^O , W_i^Q , W_i^K , W_i^V macierzami wag. Wówczas:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)W^O. \quad (3)$$

Pojedyncza głowica obliczana jest następująco:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \quad (4)$$

gdzie $i \in \{1, \dots, h\}$, z kolei

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q^\top K}{\sqrt{d}}\right)V. \quad (5)$$

Autorzy pracy *Attention Is All You Need* używają 6 bloków w enkoderze i dekoderze, $d = 64$, $h = 8$.

Masked multi-head attention

- ▶ Zamaskowana atencja wielogłowicowa (ang. masked multi-head attention) dotyczy sytuacji, w której część wartości zostaje zamaskowana, tzn. prawdopodobieństwo wyboru takich wartości wynosi 0.
- ▶ Podczas dekodowania wartość wyjściowa zależy tylko od wcześniejszych, a nie przyszłych wyjść.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q^T K}{\sqrt{d}}\right)V \quad (6)$$

W zamaskowanej atencji, w liczniku argumentu funkcji Softmax dodajemy macierz M składającą się z wartości 0 oraz $-\infty$, tj.:

$$\text{MaskedAttention}(Q, K, V) = \text{Softmax}\left(\frac{Q^T K + M}{\sqrt{d}}\right)V. \quad (7)$$

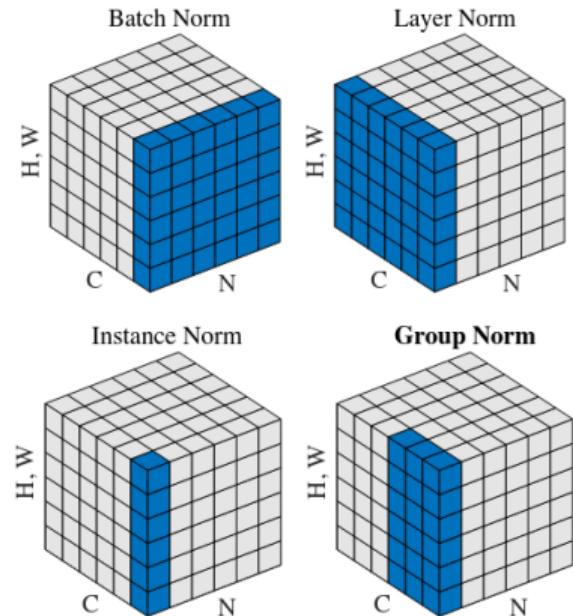
Normalizacja warstwy

- ▶ Wykonywana jest dla całej warstwy, a nie dla próbki danych, jak w przypadku normalizacji wsadowej (ang. batch normalization).
- ▶ Normalizuje wartości wejściowe do neuronów w danej warstwie, by miały średnią 0 oraz wariancję równą 1.
- ▶ Dla i -tej jednostki ukrytej, gdzie $i \in \{1, \dots, H\}$, wyliczane jest:

$$LN(h_i) = \frac{g_i}{\sigma}(h_i - \mu), \quad (8)$$

gdzie g_i jest parametrem zysku (ang. gain

parameter), $\mu = \frac{1}{H} \sum_{i=1}^H h_i$, zaś $\sigma = \sqrt{\frac{1}{H} \sum_{i=1}^H (h_i - \mu)^2}$.



Porównanie metod normalizujących. N oznacza rozmiar batcha, C to liczba kanałów, (H, W) to osie przestrzenne.

Wektory pozycyjne

- Osadzone wektory pozycyjne (ang. positional embeddings) mają za zadanie opisać pozycję danego elementu w ciągu w sposób jednoznaczny.

Index

<i>Sequence of token</i>	<i>Positional Encoding Matrix</i>
I	$P_{00} \ P_{01} \ \dots \ P_{0d}$
am	$P_{10} \ P_{11} \ \dots \ P_{1d}$
a	$P_{20} \ P_{21} \ \dots \ P_{2d}$
Robot	$P_{30} \ P_{31} \ \dots \ P_{3d}$

Źródło:
machine-
learningmys-
tery.com

Positional Encoding Matrix for the sequence 'I am a robot'

Wektory pozycyjne

- ▶ Przypuśćmy, że nasz ciąg wejściowy ma wymiar L i chcemy określić reprezentację k -tej pozycji tego ciągu, gdzie $k \in \{0, 1, \dots, L - 1\}$. Wówczas

$$P(k, 2i) = \sin\left(\frac{k}{n^{\frac{2i}{d}}}\right), \quad (9)$$

$$P(k, 2i + 1) = \cos\left(\frac{k}{n^{\frac{2i}{d}}}\right); \quad (10)$$

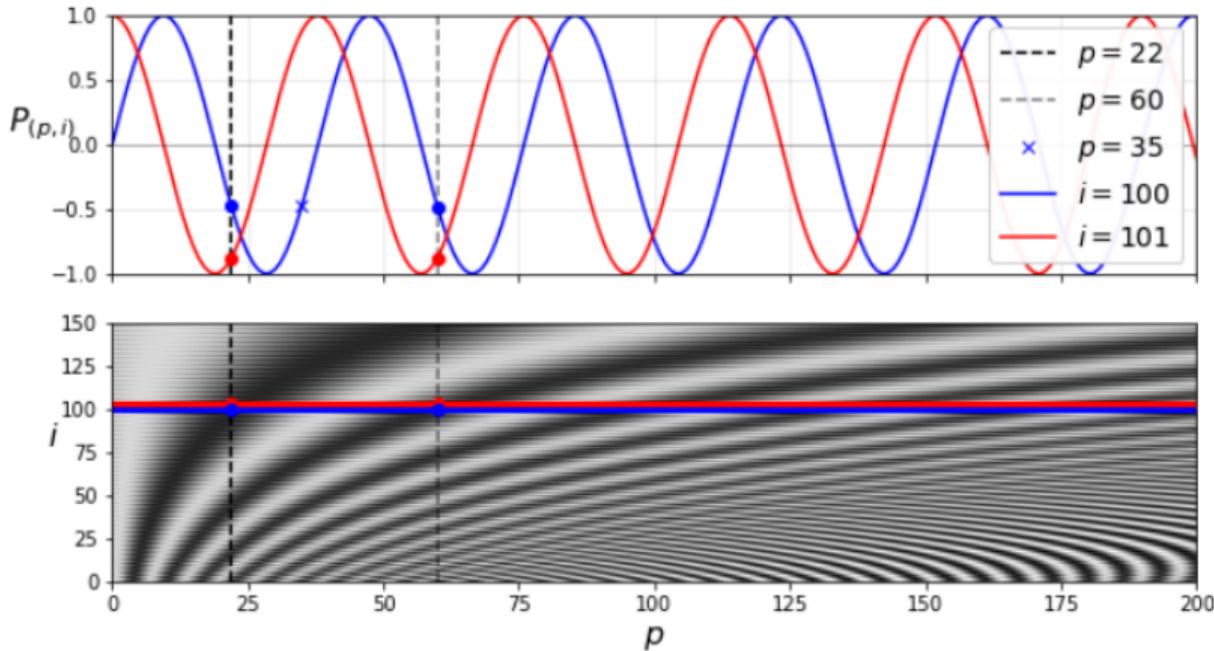
gdzie d jest wymiarem wektora pozycyjnego, $P(k, j)$ odwzorowuje k -tą pozycję wejściowego ciągu do elementu znajdującego się w pozycji (k, j) macierzy pozycyjnej, n jest parametrem (w pracy *Attention is All You Need* określonym na 10000), $i \in \{0, 1, \dots, \frac{d}{2}\}$ określa kolumnę macierzy pozycyjnej. Ustalone i odwzorowuje element z użyciem sinusa oraz cosinusa.

Wektory pozycyjne

Sequence	Index of token, k	Positional Encoding Matrix with $d=4$, $n=100$			
		$i=0$	$i=0$	$i=1$	$i=1$
I	0	$P_{00}=\sin(0) = 0$	$P_{01}=\cos(0) = 1$	$P_{02}=\sin(0) = 0$	$P_{03}=\cos(0) = 1$
am	1	$P_{10}=\sin(1/1) = 0.84$	$P_{11}=\cos(1/1) = 0.54$	$P_{12}=\sin(1/10) = 0.10$	$P_{13}=\cos(1/10) = 1.0$
a	2	$P_{20}=\sin(2/1) = 0.91$	$P_{21}=\cos(2/1) = -0.42$	$P_{22}=\sin(2/10) = 0.20$	$P_{23}=\cos(2/10) = 0.98$
Robot	3	$P_{30}=\sin(3/1) = 0.14$	$P_{31}=\cos(3/1) = -0.99$	$P_{32}=\sin(3/10) = 0.30$	$P_{33}=\cos(3/10) = 0.96$

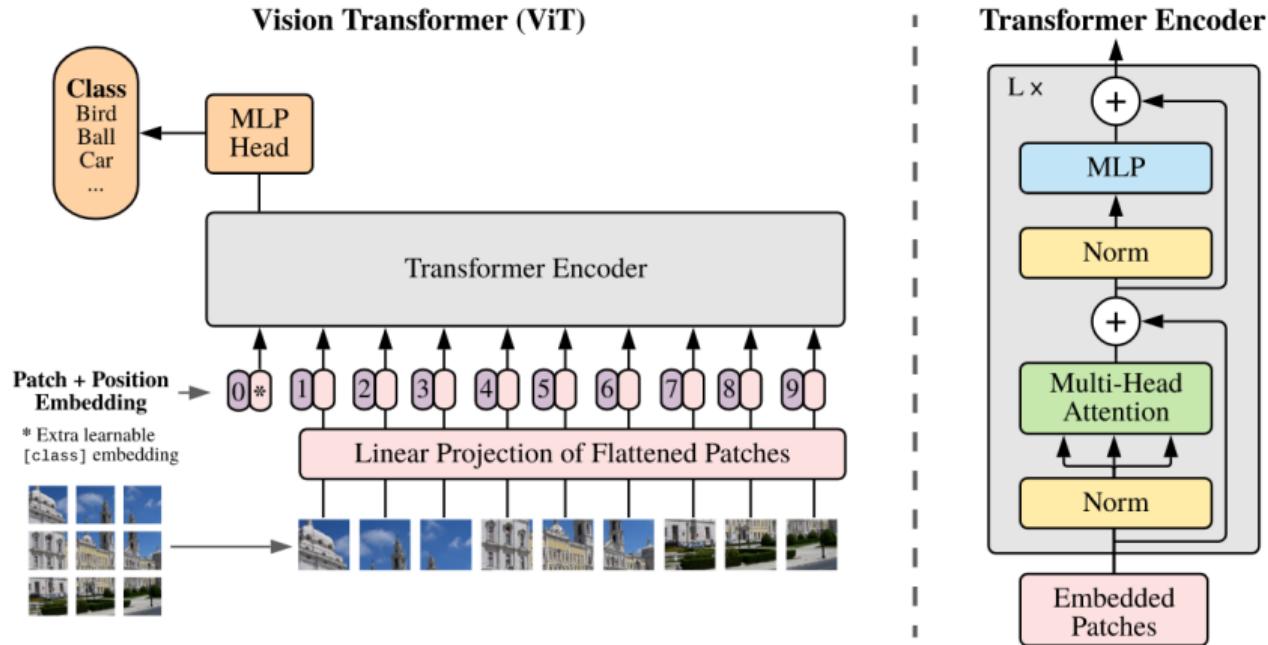
Positional Encoding Matrix for the sequence 'I am a robot'

Wektory pozycyjne



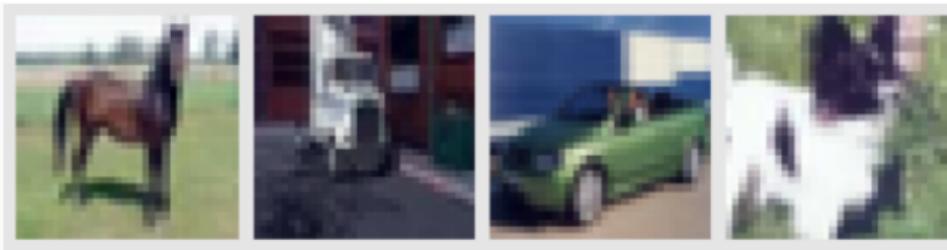
Macierz sinusoidalnych oraz cosinusoidalnych kodowań pozycyjnych. p oznacza pozycję słowa w zdaniu, i oznacza kolejny wymiar kodowania, z kolei $P(p, i)$ wartość kodowania p -tego słowa w i -tym wymiarze.

Transformator wizyjny (ang. Vision Transformer, ViT)

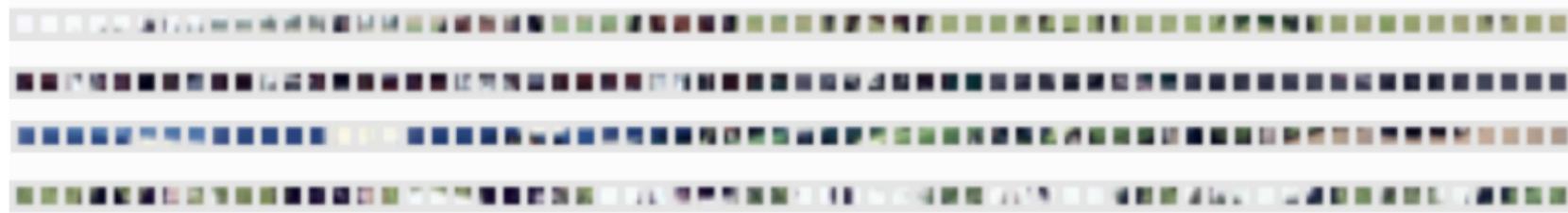


Podział obrazu na fragmenty (ang. patches)

- ▶ Przykładowe obrazy zbioru CIFAR-10:



- ▶ Powyższe obrazy jako ciągi fragmentów:

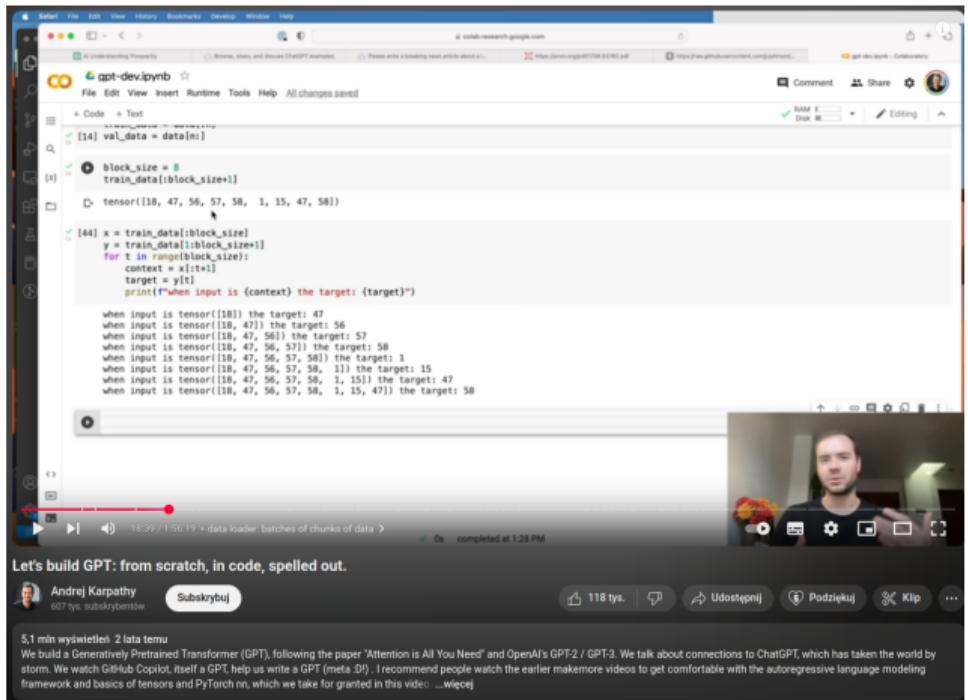


Moduły transformera wizyjnego

- ▶ Warstwa **rzutu liniowego** przekształca wejściowe fragmenty obrazu do wektora cech o większym rozmiarze. Jest to liniowa warstwa, która niezależnie przyjmuje na wejściu każdy z fragmentów.
- ▶ **Token klasyfikujący (CLS)**, który jest dodawany do przetworzonych fragmentów obrazu i ma za zadanie *reprezentować cały obraz*, co dalej może posłużyć do klasyfikacji.
- ▶ Trenowalne **wektory pozycyjne**, które są dodawane do tokenów (danych wejściowych wraz z CLSem).
- ▶ Część kodująca transformera (ang. *transformer encoder*).
- ▶ Głowa MLP.

Rekomendacje

- ▶ Tutorial z transformerów: **UvA DL Notebooks, Tutorial 6: Transformers and Multi-Head Attention.**
- ▶ Tutorial z transformerów wizyjnych – omówienie wraz z przykładem kodu: **UvA DL Notebooks, Tutorial 15: Vision Transformers.**
- ▶ Implementacja modelu językowego minGPT z wprowadzeniem Andreja Karpathy'ego (live coding) – **Let's build GPT: from scratch, in code, spelled out.**
- ▶ Publikacja: **K. He et al, Masked Autoencoders Are Scalable Vision Learners, CVPR 2022.**



Źródło: youtube.com

Dziękuję za atencję!

ŹRÓDŁA:

1. A. Vaswani et al, Attention Is All You Need, Proceedings of NeurIPS 2017.
2. P. Poupart, CS480/680 Lecture 19: Attention and Transformer Networks, University of Waterloo, 2019.
3. M. Saeed, A Gentle Introduction to Positional Encoding in Transformer Models, Part 1, machinelearningmystery.com, 2023.
4. P. Lippe, Tutorial 15: Vision Transformers, uvadlc-notebooks.readthedocs.io.