Mechanizm uwagi

Transformery

Standardowa warstwa z uwagą

Wejście: dwie sekwencje $X \in \mathbb{R}^{T \times D}$ i $X' \in \mathbb{R}^{T' \times D}$, gdzie D to wymiar reprezentacji a T i T' to długości sekwencji

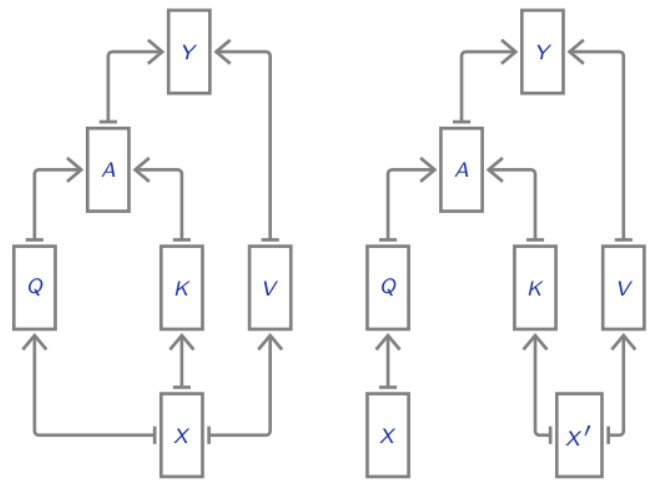
Projekcja do sekwencji zapytań Q (*queries*), K kluczy (*keys*) i V wartości (*values*)

$$Q = XW_Q \qquad K = X'W_K \qquad V = X'W_V$$

gdzie $W_Q \in \mathbb{R}^{D \times d_k}$, $W_K \in \mathbb{R}^{D \times d_k}$, $W_V \in \mathbb{R}^{D \times d_v}$

Macierz uwagi $A \in \mathbb{R}^{T \times T'}$

$$A = \text{softmax}_{\text{row}}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$$



Sekwencja wyjściowa

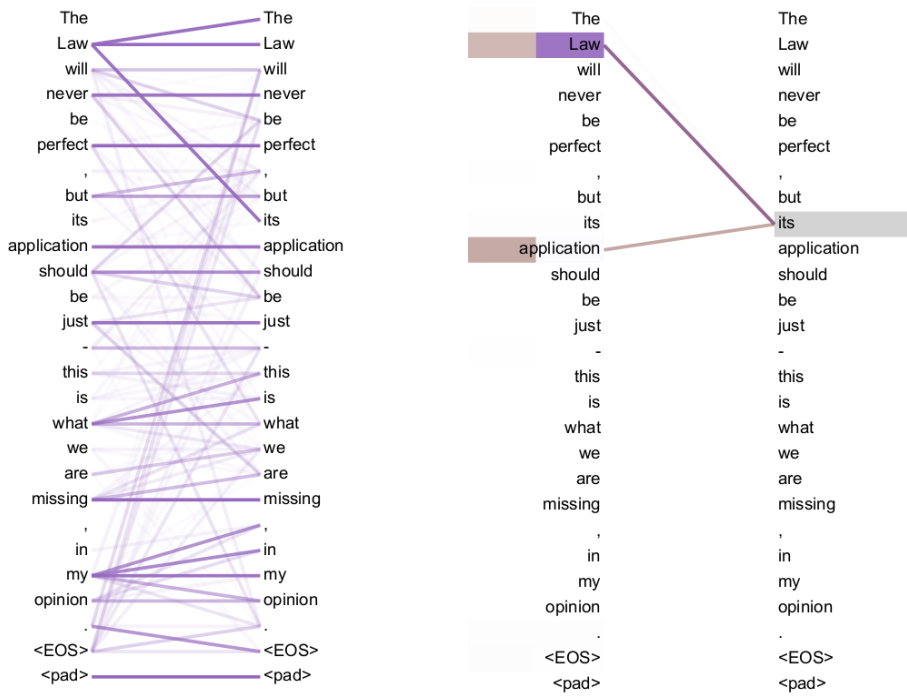
$$Y = AV$$

- samo-uwaga (*self attention*), jeżeli $X = X'$
- uwaga krzyżowa (*cross attention*), jeżeli $X \neq X'$

Uwaga to wszystko czego potrzebujesz

Uwaga (*attention*) jest uśrednieniem sekwencji **wartości** $V \in \mathbb{R}^{T' \times d_v}$ skojarzonych z sekwencją **kluczy** $K \in \mathbb{R}^{T' \times d_k}$ dopasowanych do sekwencji **zapytań** $Q \in \mathbb{R}^{T \times d_k}$

$$\text{Attention}(Q, K, V) = AV = \text{softmax}_{\text{row}}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



☞ BertViz

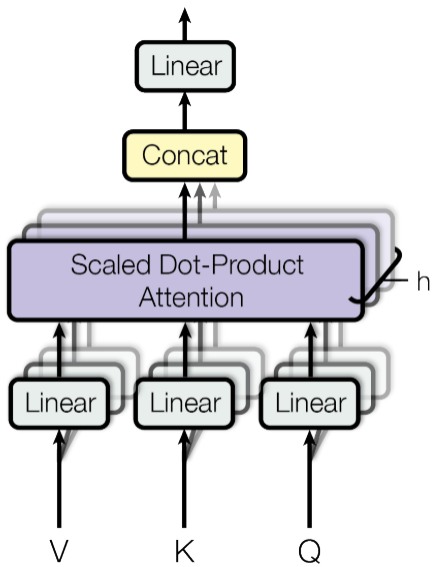
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

Uwaga o wielu głowach (*Multi-head attention*)

Uwaga z wieloma głowami pozwala skupić uwagę modelu na informacjach zawartych w różnych reprezentacjach (osobne projekcje Q_i, K_i, V_i) i w różnych pozycjach sekwencji

$$\text{MultiHead}(Q, K, V) = \text{Concat}\,(Y_1, \ldots, Y_{\text{h}})\, W_H$$

gdzie $W_H \in \mathbb{R}^{hd_v \times D}$



W przypadku standardowego transformera ustala się $d_k = d_v = D/h$, dzięki redukcji wymiaru każdej głowy całkowity koszt obliczeń pozostaje zbliżony do pojedynczej warstwy uwagi o pełnej wymiarowości

Standardowa operacja uwagi jest **niezmiennicza** względem permutacji kluczy i wartości:

$$\text{Attention}(Q, \sigma(K), \sigma(V)) = \text{Attention}(Q, K, V)$$

oraz **ekwiwariantna** względem permutacji zapytań, tzn. wynikowy tensor jest permutowany w ten sam sposób:

$$\text{Attention}(\sigma(Q), K, V) = \sigma(\text{Attention}(Q, K, V))$$

gdzie $\sigma(X)$ oznacza permutację wierszy macierzy

W konsekwencji *self-attention* i *cross-attention* są ekwiwariantne względem permutacji elementów w sekwencji X, a *cross-attention* jest niezmiennicza względem permutacji elementów w sekwencji X'.

Dlaczego samo-uwaga?

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n - długość sekwencji, d - wymiar reprezentacji, k - rozmiar filtru splotu

- warstwy *self-attention* są szybsze od warstw rekurencyjnych gdy długość sekwencji n jest mniejsza od wymiarowości reprezentacji d

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

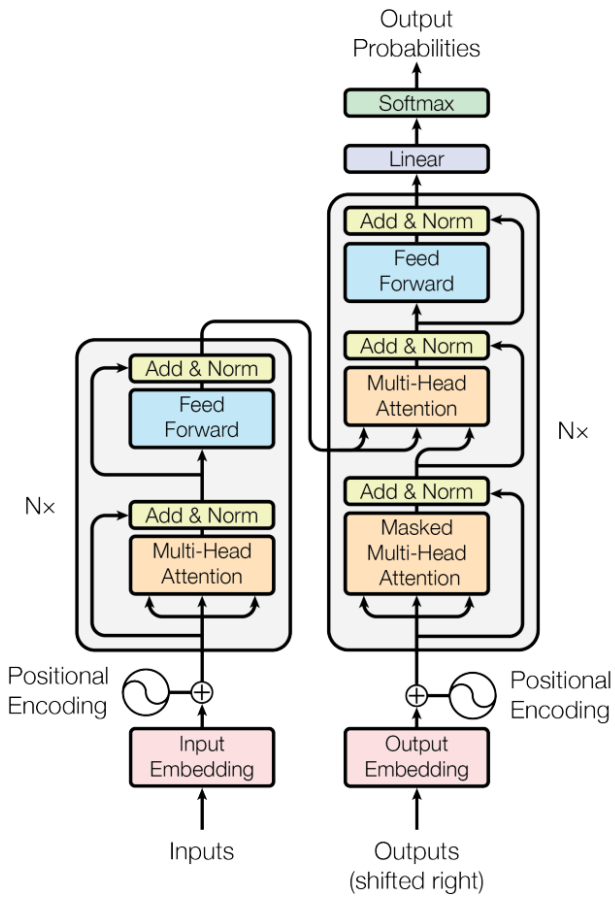
Transformer (Vaswani, 2017)

Transformery używają samo-uwagi do modelowania sekwencji w czysto jednokierunkowej sieci (bez warstw rekurencyjnych). Stosowane do budowy modeli językowych, np. BERT, GPT

Pierwotny model transformera $N = 6$
Model GPT-3 ma $N = 96$

Enkoder: bloki kodujące z samo-uwagą, połączenia skrótowe między warstwami

Dekoder: bloki dekodujące z samo-uwagą, w każdym bloku dodatkowa warstwa z uwagą krzyżową ponad wyjściem bloków enkodera, działa w trybie autoregresji - przewiduje kolejne słowo używając poprzednie słowo jako wejście



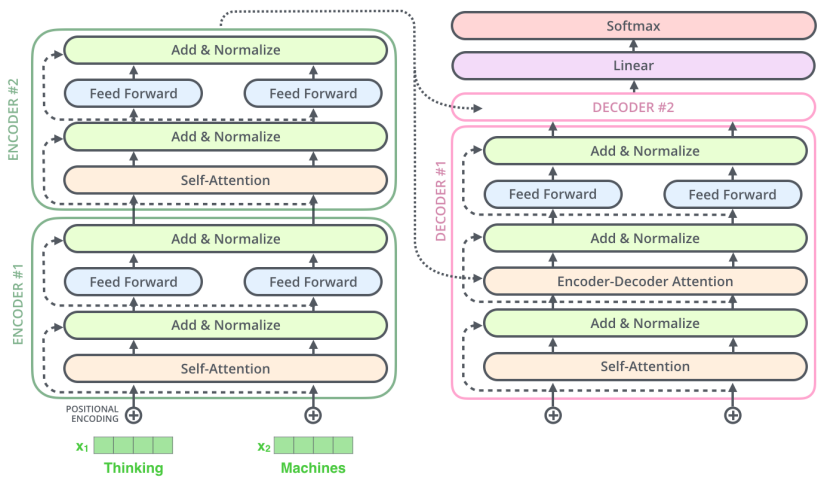
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

Transformer: kluczowe elementy

- mechanizm uwagi (*attention mechanism*) - pozwala modelowi skupić się na różnych częściach sekwencji wejściowej podczas generowania każdego elementu sekwencji wyjściowej
- warstwy z wieloma głowami uwagi (*multi-head attention layers*) - pozwalają modelowi skupić się na różnych aspektach sekwencji wejściowej jednocześnie
- warstwy feed-forward - pozwalają modelowi przetwarzać informacje na różnych poziomach abstrakcji
- normalizacja warstw (*layer normalization*) - pomaga stabilizować i przyspieszać trening modelu
- połączenia skrótowe (*residual connections*) - pomagają w propagacji gradientów i zapobiegają zanikowi gradientów w głębokich sieciach

Działanie transformera

- Wejście enkodera: sekwencja elementów (np. słów w zdaniu) w reprezentacji wektorowej (embedingi)
- Wejście dekodera: sekwencja embedingów przesunięta o jedno miejsce w prawo (tzw. *teacher forcing*) oraz sekwencja wyjściowa enkodera
- Dekoder generuje sekwencję wyjściową, przewidując kolejne elementy sekwencji wyjściowej na podstawie poprzednich elementów sekwencji wyjściowej i całej sekwencji wejściowej enkodera (zob. ☞ wizualizacja działania)

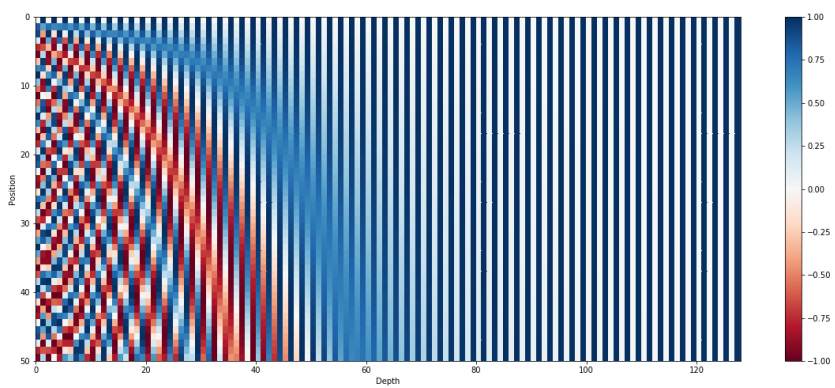


Wektor pozycyjny (Positional encoding)

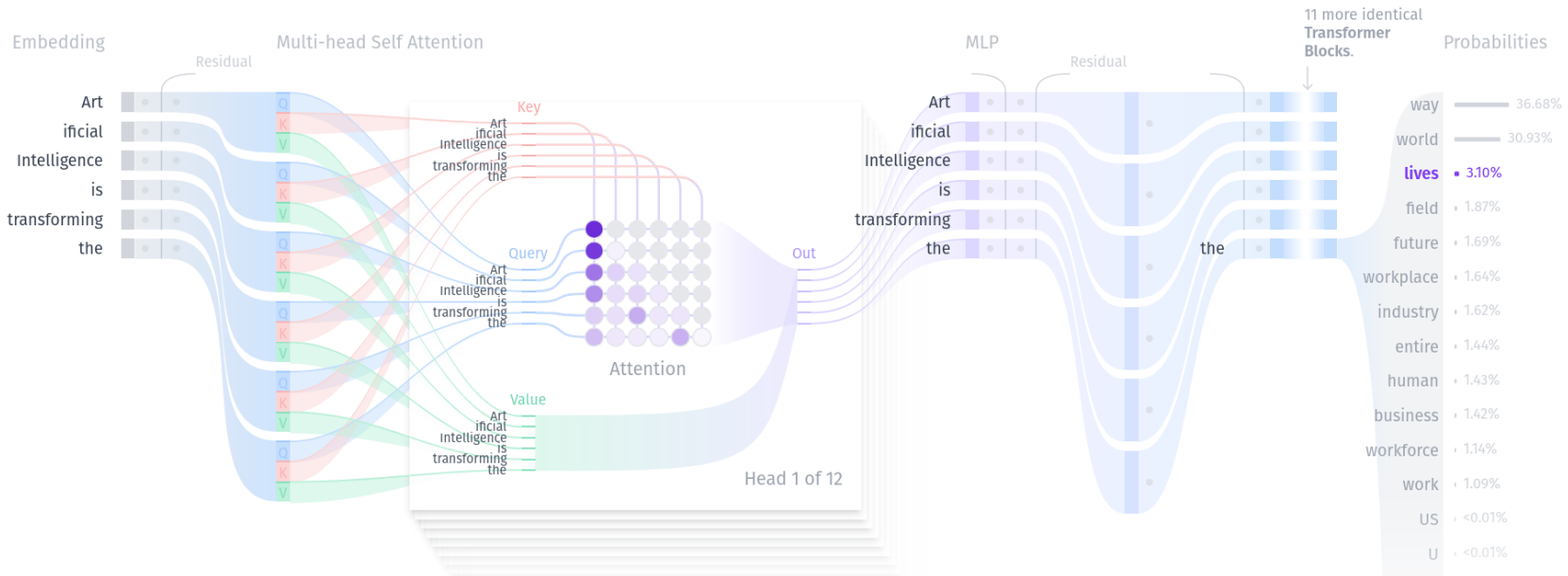
- transformer nie ma wbudowanego mechanizmu do modelowania kolejności elementów w sekwencji, więc dodaje się do wejścia enkodera i dekodera wektor pozycyjny, który jest sumowany z embedingiem elementu sekwencji
- wektor pozycyjny jest zwykle reprezentowany jako funkcja sinusoidalna o różnych częstotliwościach dla różnych wymiarów reprezentacji lub jako wektor uczony, który jest optymalizowany podczas treningu modelu
- każdy wymiar reprezentacji jest kodowany inną częstotliwością, co pozwala modelowi rozróżnić różne pozycje w sekwencji

Przykład kodowania pozycyjnego:

$$E_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$E_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$



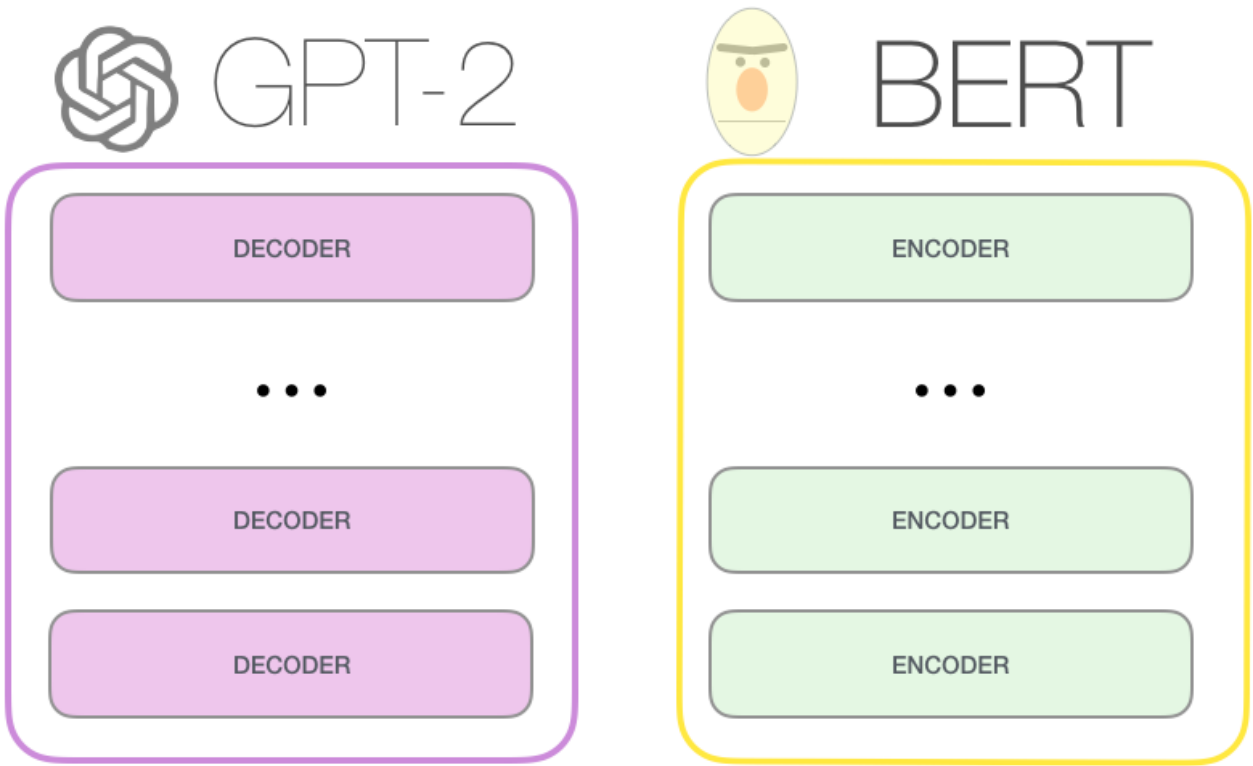
Transformer



☞ Transformer Explainer

BERT i GPT

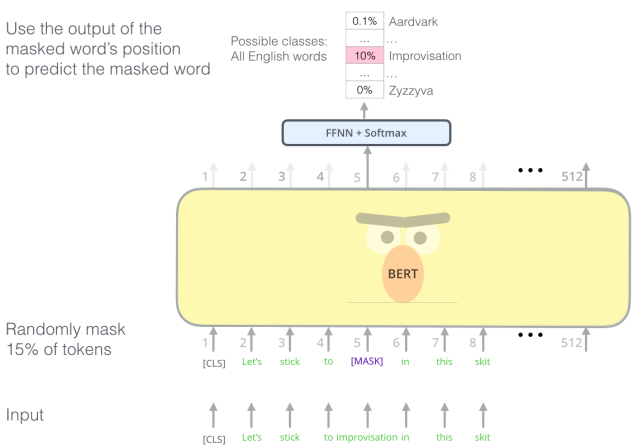
- BERT (Bidirectional Encoder Representations from Transformers) od Google (2019)
- GPT (Generative Pre-trained Transformer) od OpenAI

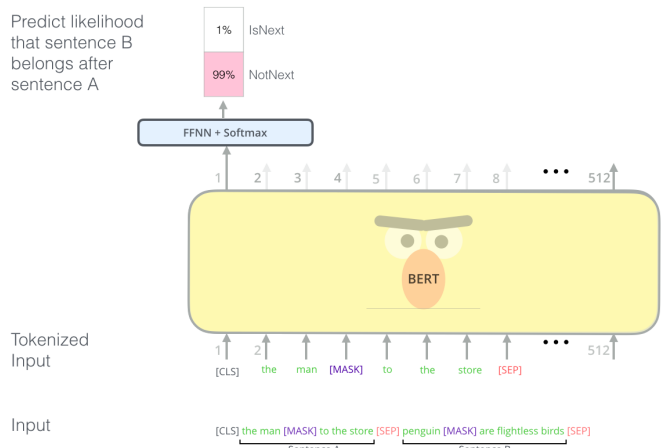


☞ *Attentional neural networks*

BERT

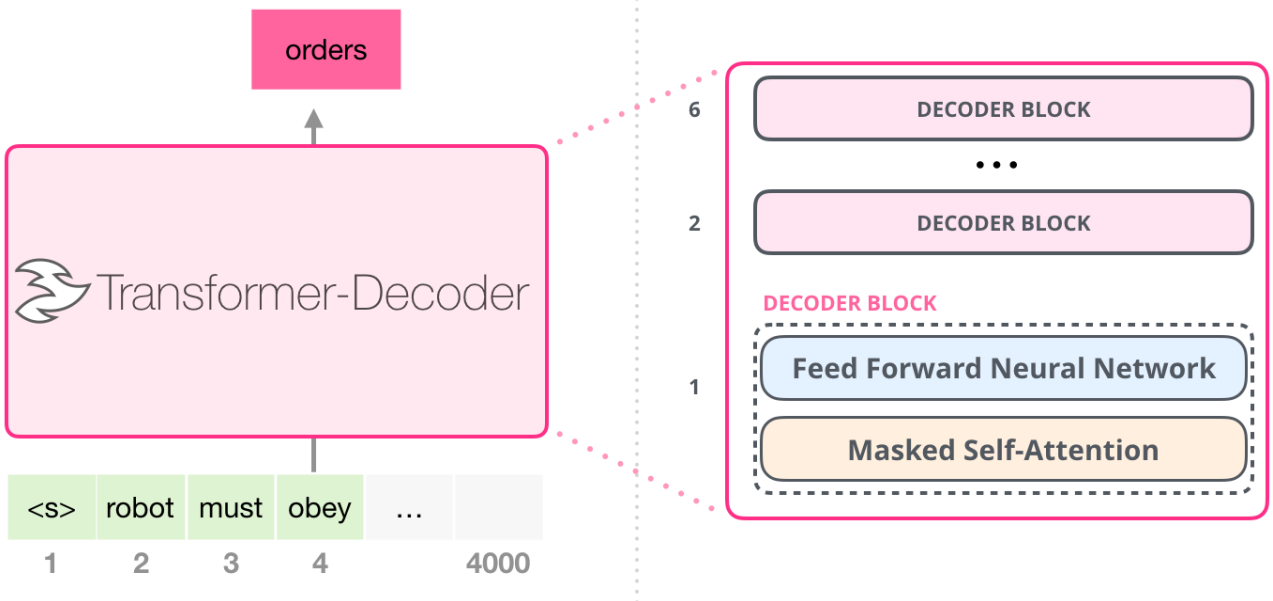
- BERT to model językowy oparty na architekturze transformera (używa tylko enkodera), zaprojektowany do zrozumienia kontekstu słów w zdaniu poprzez analizę zarówno lewego, jak i prawego kontekstu (*bidirectional*)
- maskowanie słów (*masked language modeling*), gdzie losowo maskuje się niektóre słowa w zdaniu i model musi przewidzieć, jakie to słowa
- przewidywanie relacji między zdaniami (*next sentence prediction*), gdzie model musi określić, czy dwa zdania są ze sobą powiązane





GPT

- GPT to model językowy oparty na architekturze transformera (używa tylko dekodera), zaprojektowany do generowania tekstu poprzez analizę kontekstu słów w zdaniu, ale tylko z lewej strony (*unidirectional*)
- model jest trenowany w trybie autoregresyjnym, gdzie przewiduje kolejne słowo w zdaniu na podstawie poprzednich słów



Visual Transformer (ViT)

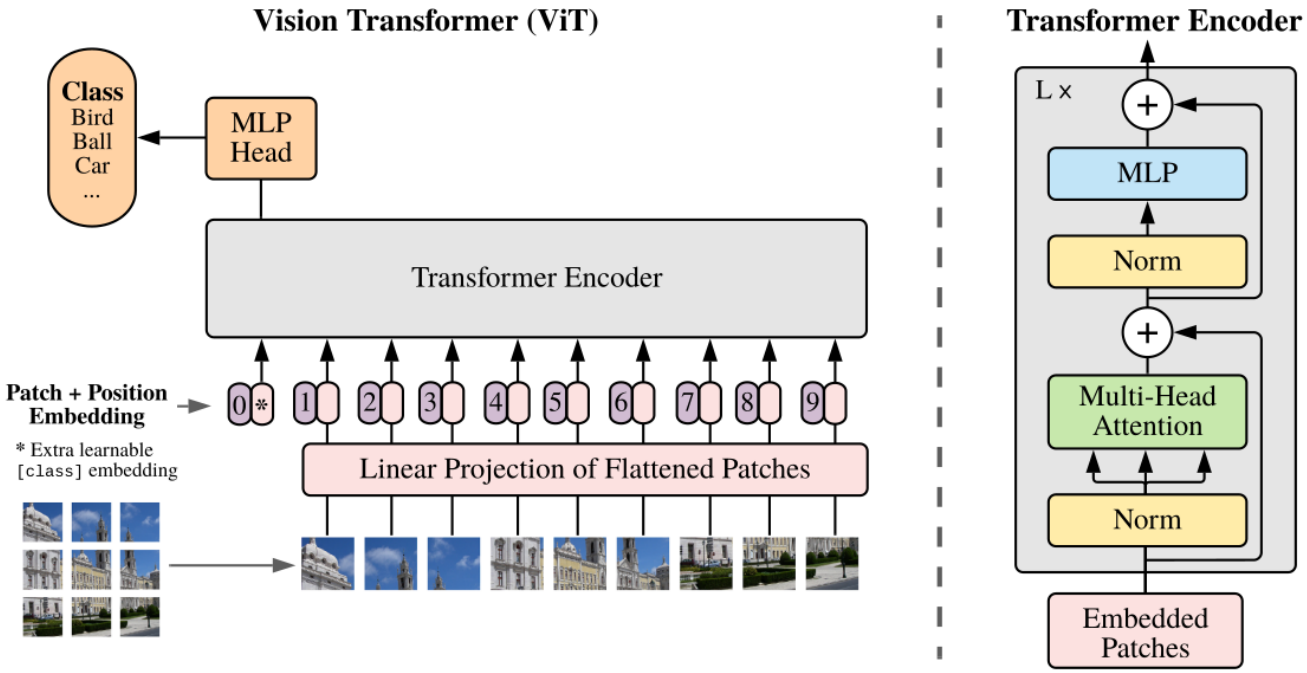


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021.

ViT

Wejście: sekwencja N łatek obrazu o rozmiarze $P \times P$ pikseli, gdzie $N = HW/P^2$ (dla obrazu o rozmiarze $H \times W$) i C kanałach (np. 3 dla RGB)

$$\mathbf{x} \in \mathbb{R}^{H \times W \times C} \longrightarrow \mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$$

gdzie $N = HW/P^2$

Liniowa transformacja do reprezentacji D wymiarowej: (patch embeddings)

$$\mathbf{z}_0 = \left[\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{pos}$$

Enkoder transformera

$$\mathbf{z}'_\ell = \text{MSA}\left(\text{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L$$

$$\mathbf{z}_\ell = \text{MLP}\left(\text{LN}\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L$$

Głowa MLP klasyfikująca

$$\mathbf{y} = \text{LN}\left(\mathbf{z}_L^0\right)$$

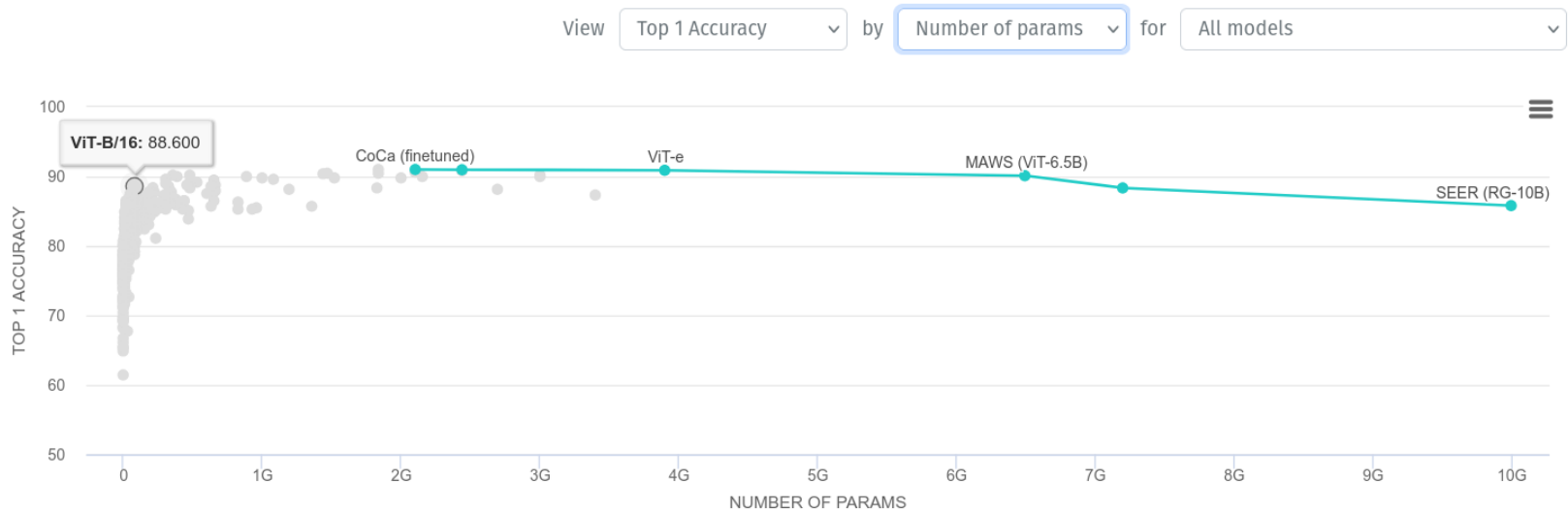
Image Classification: state of the art

Trend	Dataset	Best Model	Paper	Code	Compare
	ImageNet	CoCa (finetuned)			See all
	CIFAR-10	ViT-H/14			See all
	CIFAR-100	EffNet-L2 (SAM)			See all
	STL-10	VIT-L/16 (Spinal FC)			See all
	ObjectNet	CoCa			See all
	MNIST	Branching/Merging CNN + Homogeneous Vector Capsules			See all
	iNaturalist 2018	OmniVec2			See all
	SVHN	E2E-M3			See all
	ImageNet ReaL	Baseline (ViT-G/14)			See all
	Clothing1M	LRA-diffusion (CC)			See all

☞ Browse State-of-the-Art

Image Classification on ImageNet

Leaderboard | Community Models | Dataset



☞ Image classification on Imagenet