



# Дослідження і проектування інтелектуальних систем

Лекція 2: Увага та трансформери

Кочура Юрій Петрович  
[iuriy.kochura@gmail.com](mailto:iuriy.kochura@gmail.com)  
[@y\\_kochura](https://twitter.com/y_kochura)

# Сьогодні

Увага – це все, що Вам потрібно!

- 🔊 Кодер-декодер
- 🔊 Механізм уваги Бахданау (адитивна увага)
- 🔊 Шар уваги
- 🔊 Трансформери

# Кодер-декодер

У багатьох прикладних задачах обробка сигналів з послідовною структурою є необхідною.

- Класифікація послідовностей:
  - аналіз емоцій у тексті
  - розпізнавання активності/дій у відео
  - класифікація послідовностей ДНК
- Синтез послідовностей:
  - синтез тексту
  - синтез музики
  - синтез руху
- Переклад послідовностей:
  - розпізнавання мови
  - переклад тексту
  - прогнозування часових рядів

Нехай  $\mathcal{X}$  – це множина токенів. Тоді  $S(\mathcal{X})$  – це множина усіх послідовностей довільної довжини,

$$S(\mathcal{X}) = \{(x_1, x_2, \dots, x_n) \mid n \geq 1, x_i \in \mathcal{X}\} = \bigcup_{n=1}^{\infty} \mathcal{X}^n,$$

тоді ми формально визначаємо:

Класифікація послідовностей

$$f : S(\mathcal{X}) \rightarrow \Delta^C$$

Синтез послідовностей

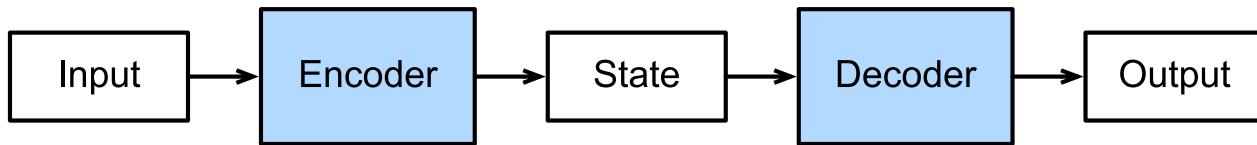
$$f : \mathbb{R}^d \rightarrow S(\mathcal{X})$$

Послідовність-послідовність

$$f : S(\mathcal{X}) \rightarrow S(\mathcal{Y})$$

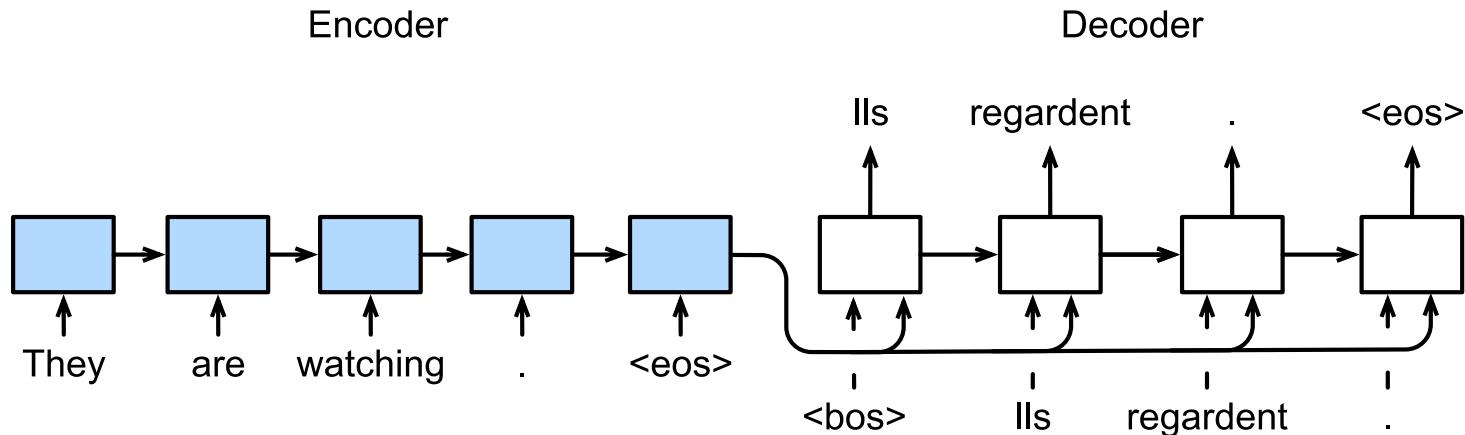
Токен – це елемент послідовності, який модель обробляє як окрему одиницю.

Коли вхідними даними є послідовність  $\mathbf{x} \in S(\mathbb{R}^p)$  змінної довжини, зазвичай використовують архітектуру «кодер–декодер», яка спочатку стискає вхідні дані в єдиний вектор  $v$ , а потім використовує його для генерації вихідної послідовності.



## Приклад

- Вхід: I am learning AI.
- Вихід: Я вивчаю ШІ.

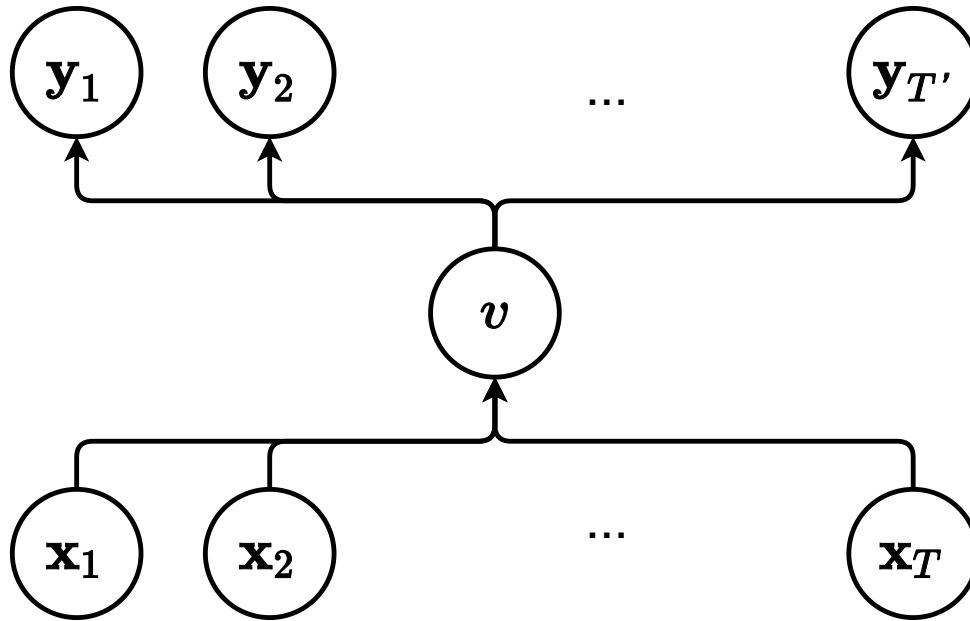


Рекурентні моделі кодер-декодер стискають вхідну послідовність  $\mathbf{x}_{1:T}$  в єдиний вектор  $v$ , а потім генерують вихідну послідовність  $\mathbf{y}_{1:T'}$  з авторегресійної генеративної моделі:

$$\mathbf{h}_t = \phi(\mathbf{x}_t, \mathbf{h}_{t-1})$$

$$v = \mathbf{h}_T$$

$$\mathbf{y}_i \sim p(\cdot | \mathbf{y}_{1:i-1}, v).$$



Ця архітектура передбачає, що єдиний вектор  $v$  містить достатньо інформації для генерації всієї вихідної послідовності. Це часто є **складним завданням** для довгих послідовностей.

# Механізм уваги Бахданау (адитивна увага)

# NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

**Dzmitry Bahdanau**

Jacobs University Bremen, Germany

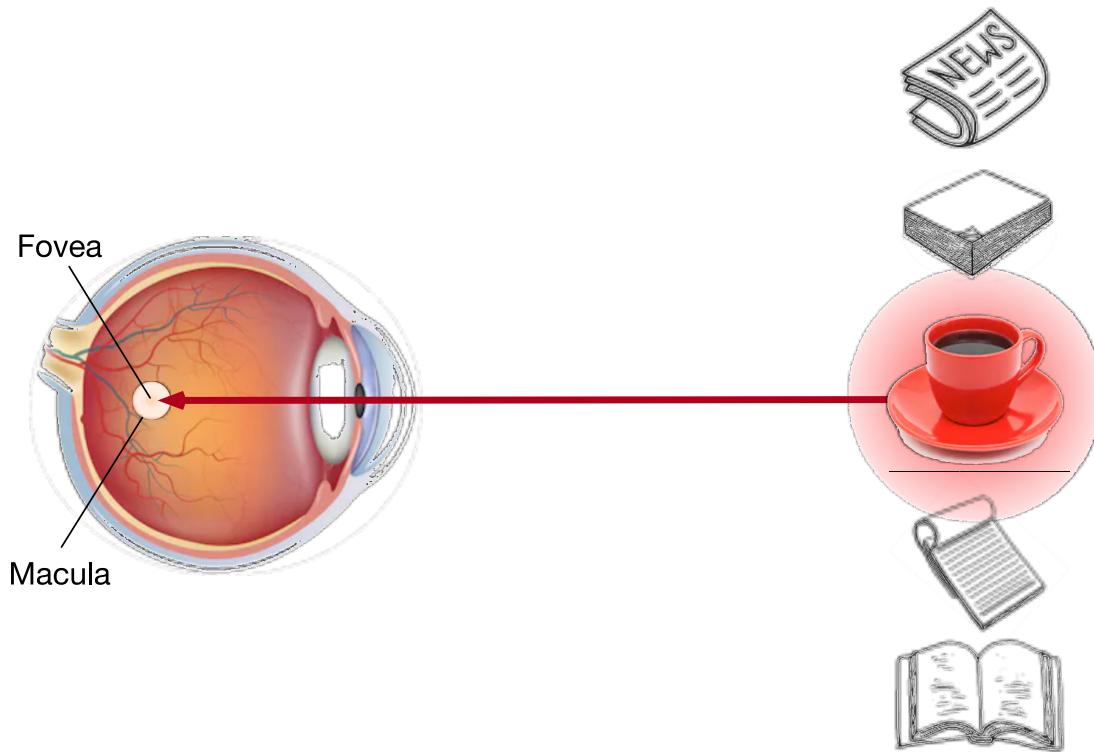
**KyungHyun Cho    Yoshua Bengio\***

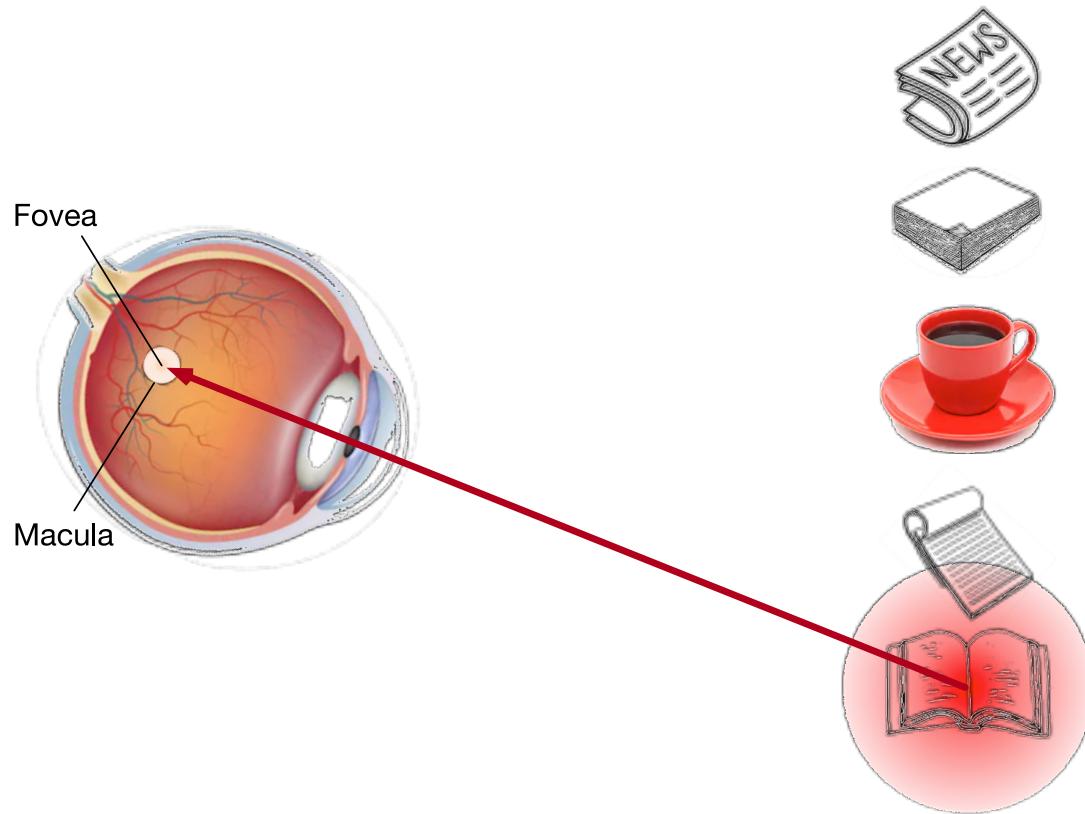
Université de Montréal

## ABSTRACT

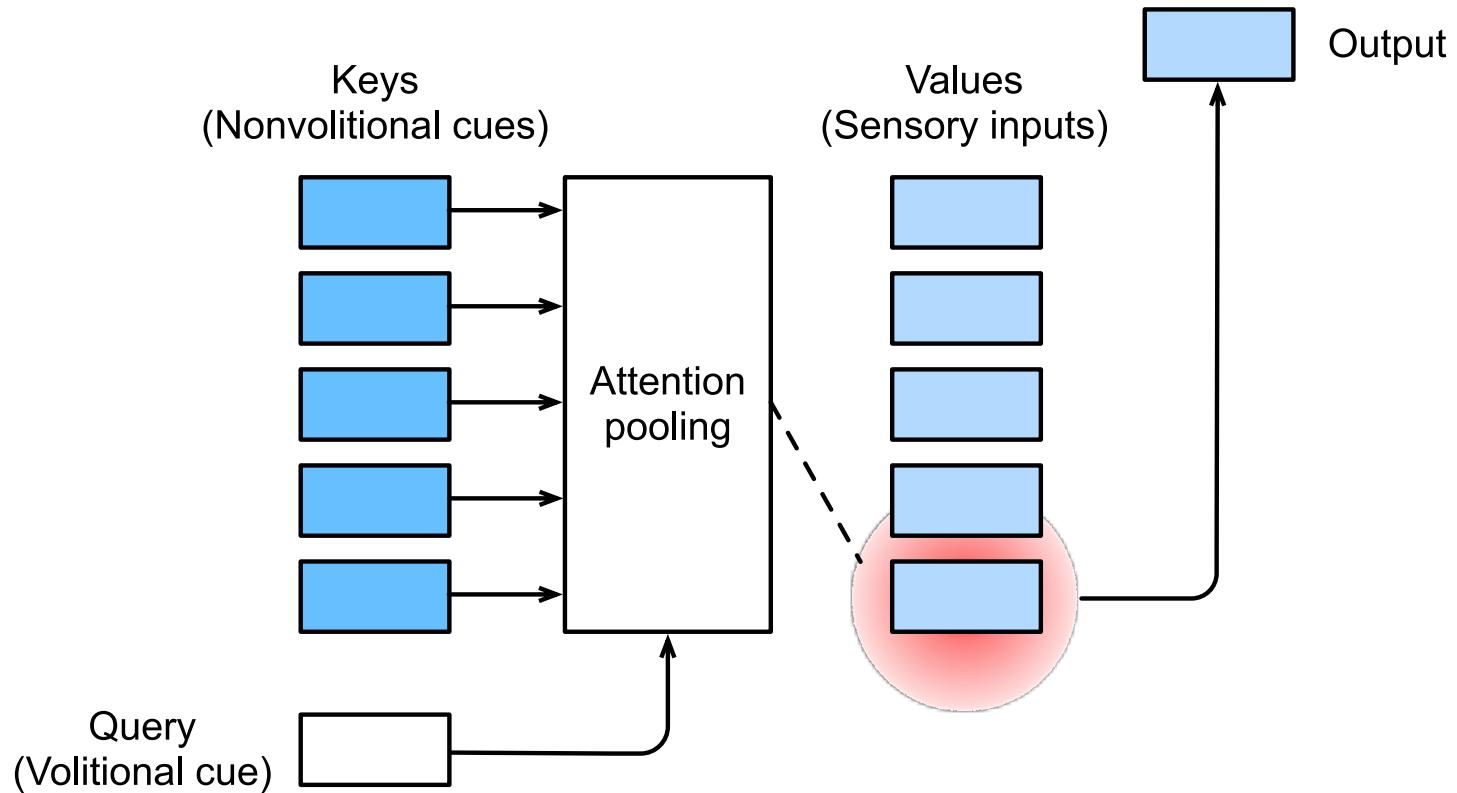
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.





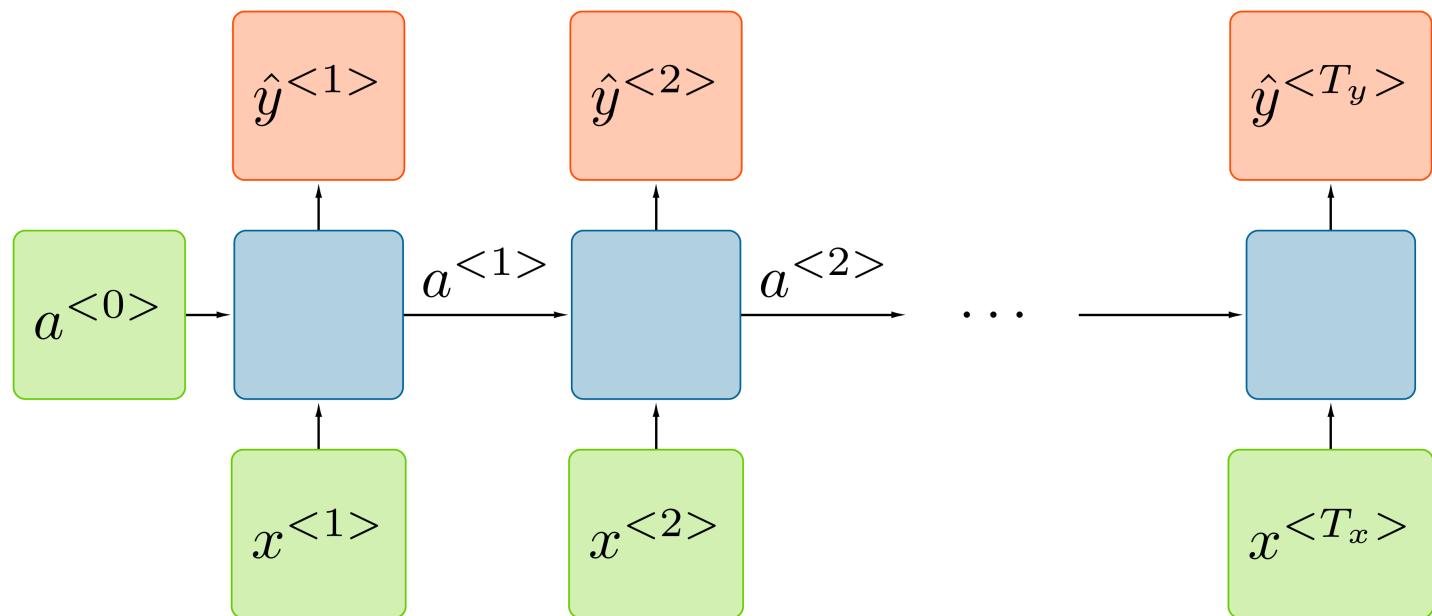


Використовуючи вольовий сигнал (бажання прочитати книгу), який залежить від завдання, увага спрямовується на книгу під вольовим контролем.

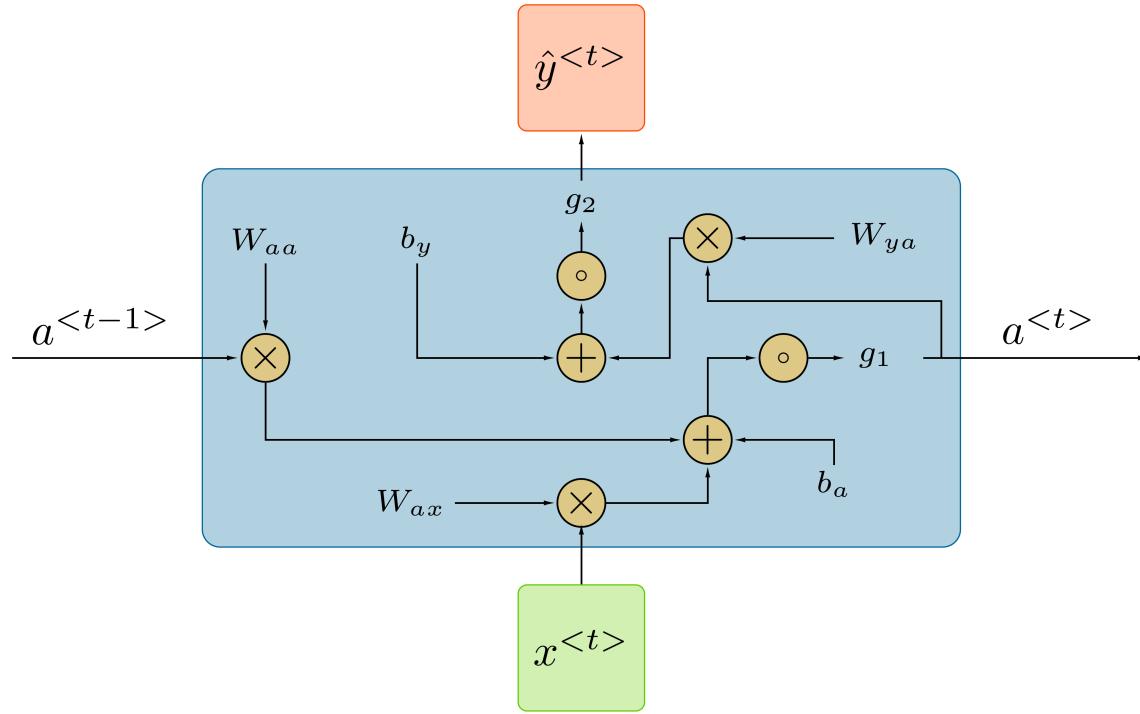


Механізм уваги може переносити інформацію з окремих частин вхідного сигналу до частин виходу, визначених динамічно. Тобто модель на кожному кроці сама «вирішує», на які токени вхідної послідовності звернути увагу для формування поточного вихідного токена.

За припущенням, що кожен вихідний токен утворюється з одного або кількох вхідних токенів, декодер має зосереджуватися лише на тих токенах, які є релевантними для генерування наступного вихідного токена.

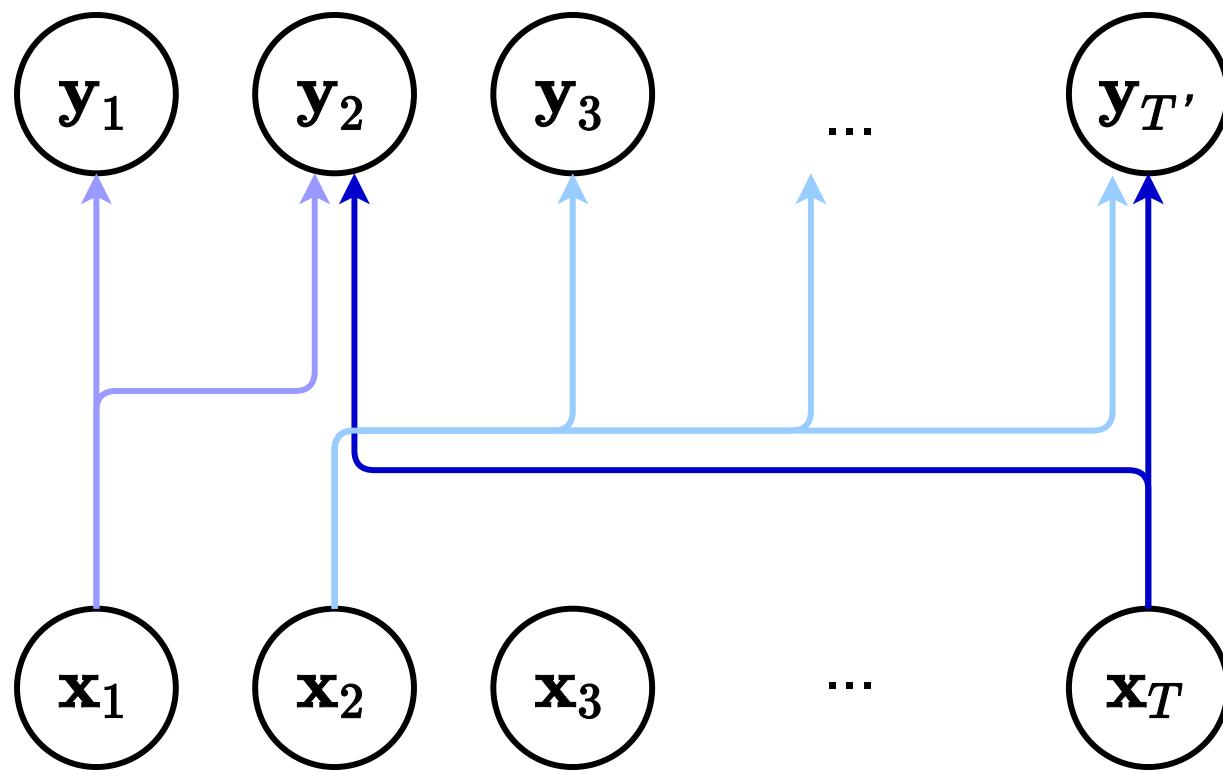


# Пряме поширення



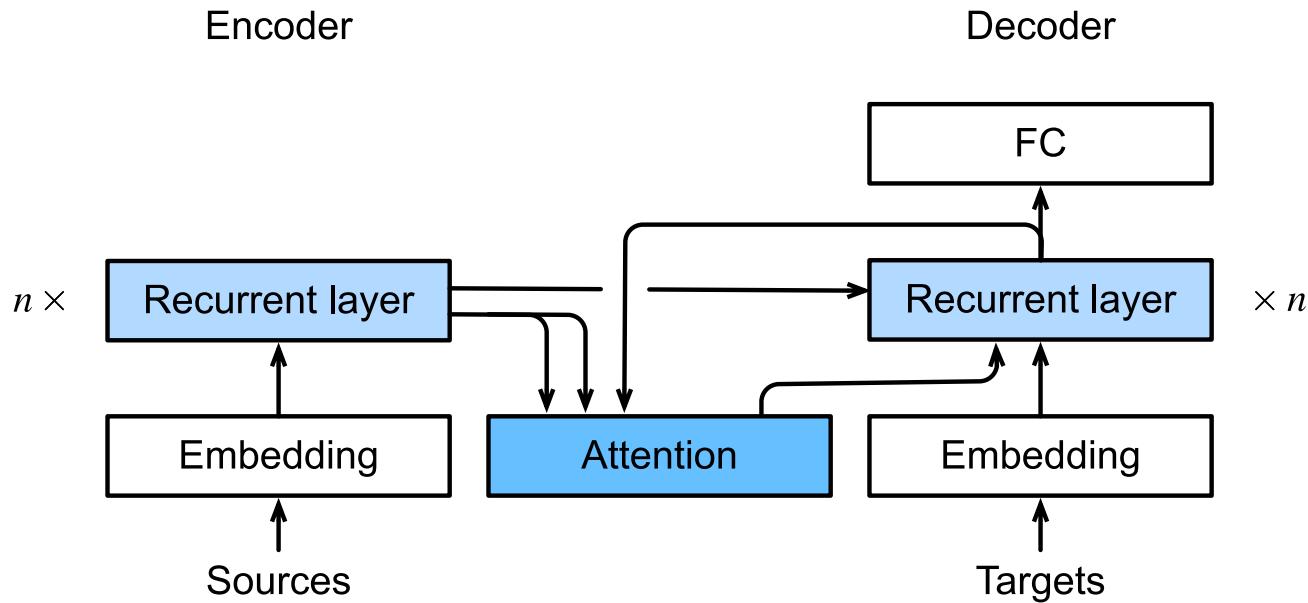
Для кожного часового кроку  $t$  активація  $a^{<t>}$  і вихід  $y^{<t>}$  виражуються таким чином:

$$\begin{aligned} a^{<t>} &= g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \\ \hat{y}^{<t>} &= g_2(W_{ya}a^{<t>} + b_y) \end{aligned}$$



## Машинний переклад на основі уваги

Та сама архітектура RNN в якій **кодер** та **декодер** з'єднані через механізм уваги.



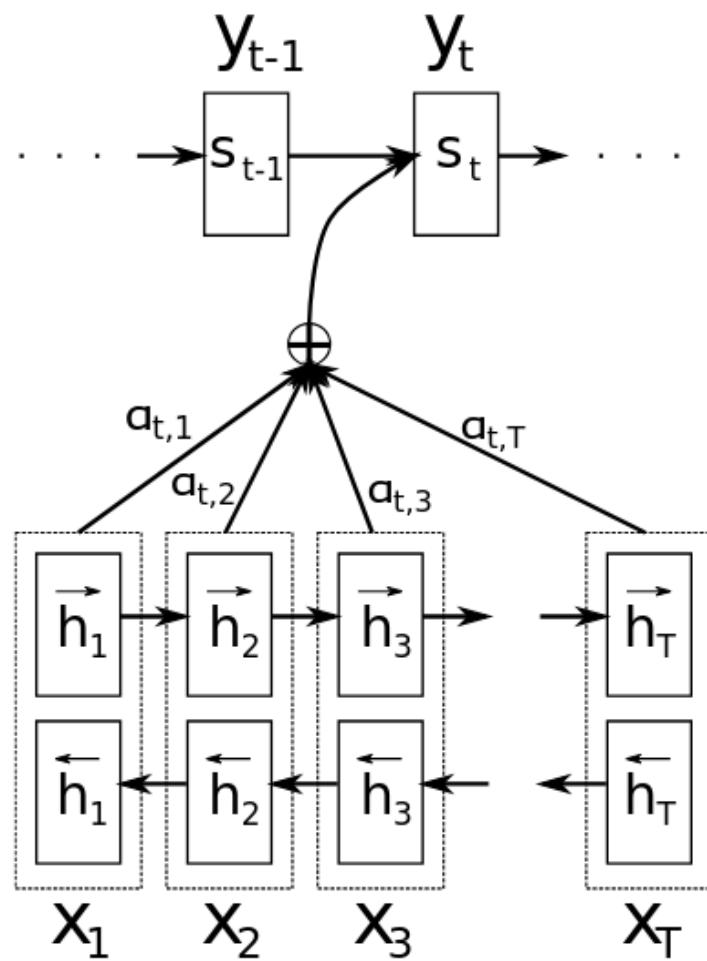
Відповідно до Bahdanau et al. (2014), кодер визначається як двонаправлена рекурентна нейронна мережа (RNN), яка обчислює вектор анотації для кожного вхідного токена,

$$\mathbf{h}_j = (\vec{\mathbf{h}}_j, \hat{\mathbf{h}}_j)$$

для  $j = 1, \dots, T$ , де  $\vec{\mathbf{h}}_j$  та  $\hat{\mathbf{h}}_j$  відповідно позначають прямі та зворотні приховані рекурентні стани (активації) двонаправленої RNN.

На основі цього декодер обчислює новий процес  $\mathbf{s}_i, i = 1, \dots, T'$ , який розглядає зважені середні значення  $\mathbf{h}_j$ , де ваги є функціями сигналу.

$\mathbf{s}_i$  — це прихований стан декодера.



Для заданих  $\mathbf{y}_1, \dots, \mathbf{y}_{i-1}$  та  $\mathbf{s}_1, \dots, \mathbf{s}_{i-1}$  спочатку обчисліть вектор уваги:

$$\alpha_{i,j} = \text{softmax}_j(a(\mathbf{s}_{i-1}, \mathbf{h}_j)) = \frac{\exp(e_{ij})}{\sum_{k=1}^T \exp(e_{ik})}$$

для  $j = 1, \dots, T$ , де

- $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ ;
- $a$  – функція оцінки уваги, яка тут задана як MLP з одним прихованим шаром  $\tanh$ ;
- $\mathbf{y}$  – це правильні вихідні токени (мітки) під час навчання;
- $\mathbf{s}$  – це приховані стани декодера, які оновлюються крок за кроком.

Потім обчисліть вектор контексту з урахуванням зважених значень  $\mathbf{h}_j$ :

$$\mathbf{c}_i = \sum_{j=1}^T \alpha_{i,j} \mathbf{h}_j.$$

Тепер модель може зробити прогноз  $\mathbf{y}_i$ :

$$\begin{aligned}\mathbf{s}_i &= f(\mathbf{s}_{i-1}, y_{i-1}, c_i) \\ \mathbf{y}_i &\sim g(\mathbf{y}_{i-1}, \mathbf{s}_i, \mathbf{c}_i),\end{aligned}$$

де  $f$  – Gated Recurrent Unit (GRU).

Це **увага до контексту**, де  $\mathbf{s}_{i-1}$  визначає, що шукати в  $\mathbf{h}_1, \dots, \mathbf{h}_T$  для обчислення  $\mathbf{s}_i$  та зразка  $\mathbf{y}_i$ .

L' accord sur la zone économique européenne a été signé en août 1992.

The agreement on the European Economic Area was signed in August 1992.

<end>

(a)

Il convient de noter que l'environnement marin est le moins connu de l'environnement.

It should be noted that the marine environment is the least known of environments.

<end>

(b)

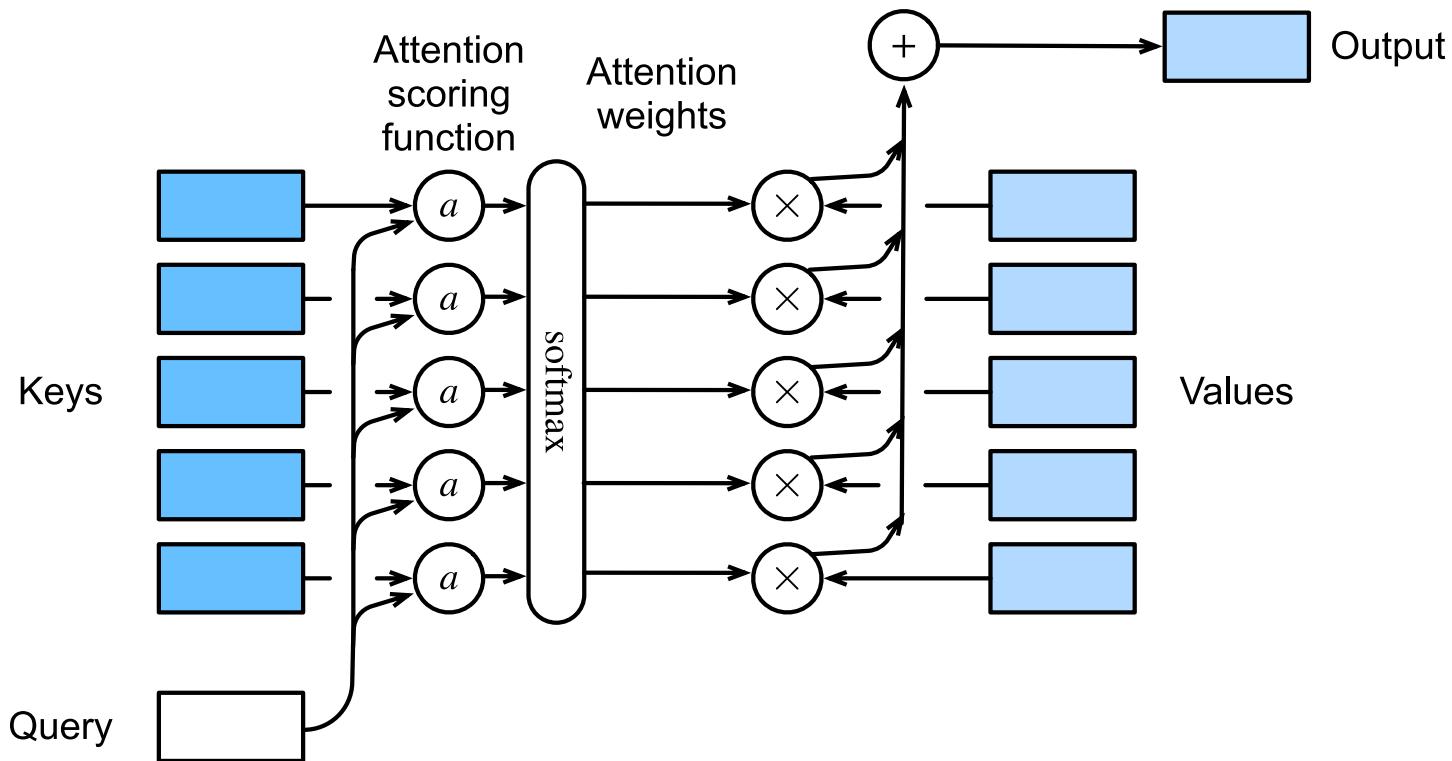
# Шар уваги

Механізми уваги можна загалом визначити наступним чином.

Для заданого контексту або вектора запиту (Query)  $\mathbf{q} \in \mathbb{R}^q$ , тензора ключів (Key)  $\mathbf{K} \in \mathbb{R}^{m \times k}$  та тензора значень  $\mathbf{V} \in \mathbb{R}^{m \times v}$  шар уваги обчислює вектор виходу  $\mathbf{y} \in \mathbb{R}^v$  за формулою:

$$\mathbf{y} = \sum_{i=1}^m \text{softmax}_i(a(\mathbf{q}, \mathbf{K}_i; \theta)) \mathbf{V}_i,$$

де  $a : \mathbb{R}^q \times \mathbb{R}^k \rightarrow \mathbb{R}$  – скалярна функція оцінки уваги,  $\theta$  – навчальні параметри, які перетворюють Query і Key у числову оцінку уваги.



## Адитивна увага

У випадку, коли запити та ключі є векторами різної довжини, ми можемо використовувати адитивну увагу як функцію оцінки.

Для  $\mathbf{q} \in \mathbb{R}^q$  та  $\mathbf{k} \in \mathbb{R}^k$  функція оцінки **адитивної уваги** має вигляд:

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^T \tanh(\mathbf{W}_q^T \mathbf{q} + \mathbf{W}_k^T \mathbf{k})$$

де  $\mathbf{w}_v \in \mathbb{R}^h$ ,  $\mathbf{W}_q \in \mathbb{R}^{q \times h}$  та  $\mathbf{W}_k \in \mathbb{R}^{k \times h}$  – навчальні параметрами.

## Масштабований скалярний добуток уваги

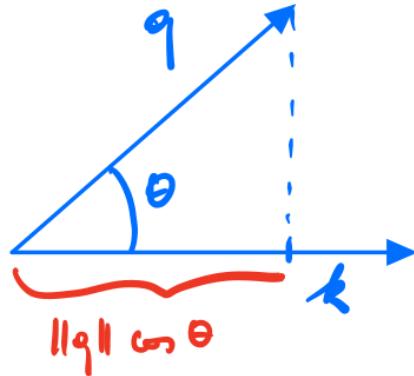
Коли запити та ключі є векторами однакової довжини  $d$ , ми можемо використовувати масштабований скалярний добуток уваги як функцію оцінки.

Для  $\mathbf{q} \in \mathbb{R}^d$  та  $\mathbf{k} \in \mathbb{R}^d$  функція оцінки **масштабованого скалярного добутку уваги** має вигляд:

$$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\sqrt{d}}$$

Для  $n$  запитів  $\mathbf{Q} \in \mathbb{R}^{n \times d}$ , ключів  $\mathbf{K} \in \mathbb{R}^{m \times d}$  і значень  $\mathbf{V} \in \mathbb{R}^{m \times v}$  шар масштабованого скалярного добутку уваги обчислює вихідний тензор:

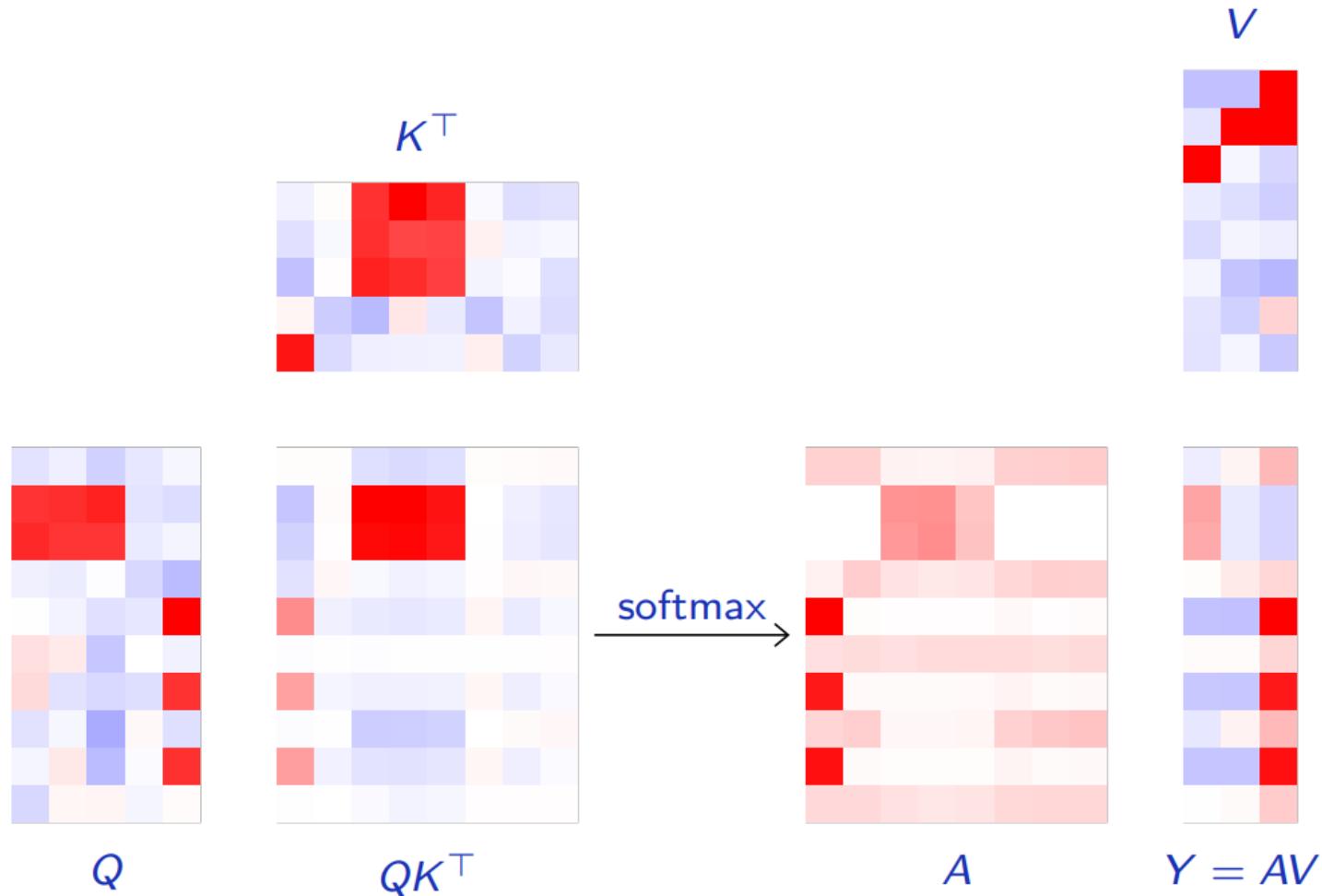
$$\mathbf{Y} = \underbrace{\text{softmax} \left( \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} \right)}_{\text{матриця уваги } \mathbf{A}} \mathbf{V} \in \mathbb{R}^{n \times v}$$



$$q^T k = \|q\| \|k\| \cos \theta$$

Скалярний добуток двох векторів – це число, яке рівне добутку довжин цих векторів на косинус кута між ними (коєфіцієнт подібності двох не нульових векторів).

Отже, матриця  $QK^T$  є **матрицею подібності** між запитами та ключами.



У сучасних стандартних моделях обробки послідовностей запити, ключі та значення є лінійними функціями вхідних даних.

Нехай матриці навчальних параметрів мають вигляд  $\mathbf{W}_q \in \mathbb{R}^{d \times x}$ ,  $\mathbf{W}_k \in \mathbb{R}^{d \times x'}$ , і  $\mathbf{W}_v \in \mathbb{R}^{v \times x'}$ , а також дві послідовності вхідних ознак  $\mathbf{X} \in \mathbb{R}^{n \times x}$  і  $\mathbf{X}' \in \mathbb{R}^{m \times x'}$ , з яких формуються запити (Queries), ключі (Keys) та значення (Values), тоді маємо:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q^T \in \mathbb{R}^{n \times d}$$

$$\mathbf{K} = \mathbf{X}'\mathbf{W}_k^T \in \mathbb{R}^{m \times d}$$

$$\mathbf{V} = \mathbf{X}'\mathbf{W}_v^T \in \mathbb{R}^{m \times v}$$

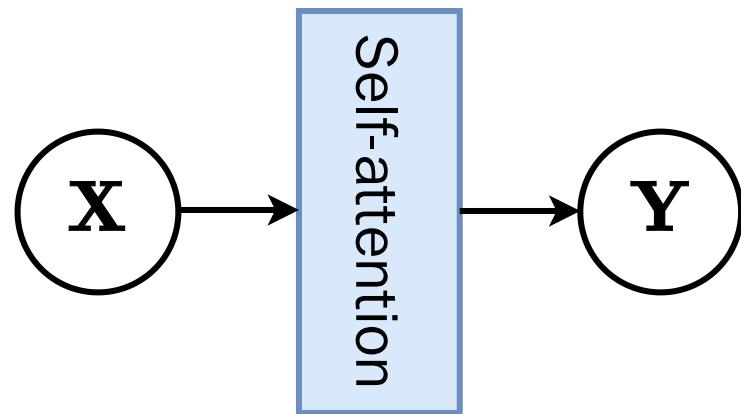
- $\mathbf{X}$  – приховані стани декодера (формують Queries).
- $\mathbf{X}'$  – приховані стани кодера (формують Keys і Values).

## Self-attention (самоувага)

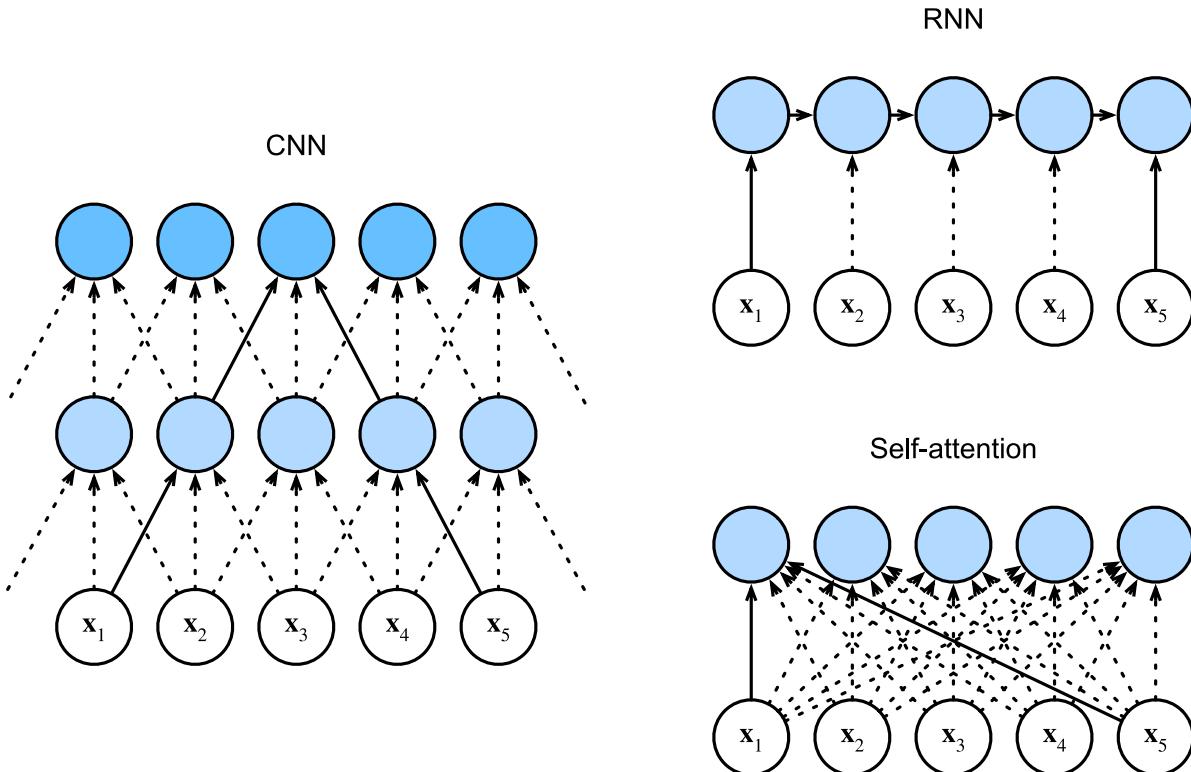
Коли запити, ключі та значення отримуються з одних і тих самих вхідних даних, механізм уваги називається **самоувагою**.

Для масштабованого скалярного добутку уваги шар самоуваги отримується, коли  $\mathbf{X} = \mathbf{X}'$ .

Тому самоувага може використовуватися як звичайний шар прямого поширення, аналогічно до повнозв'язних або згорткових шарів.



## CNNs vs. RNNs vs. self-attention

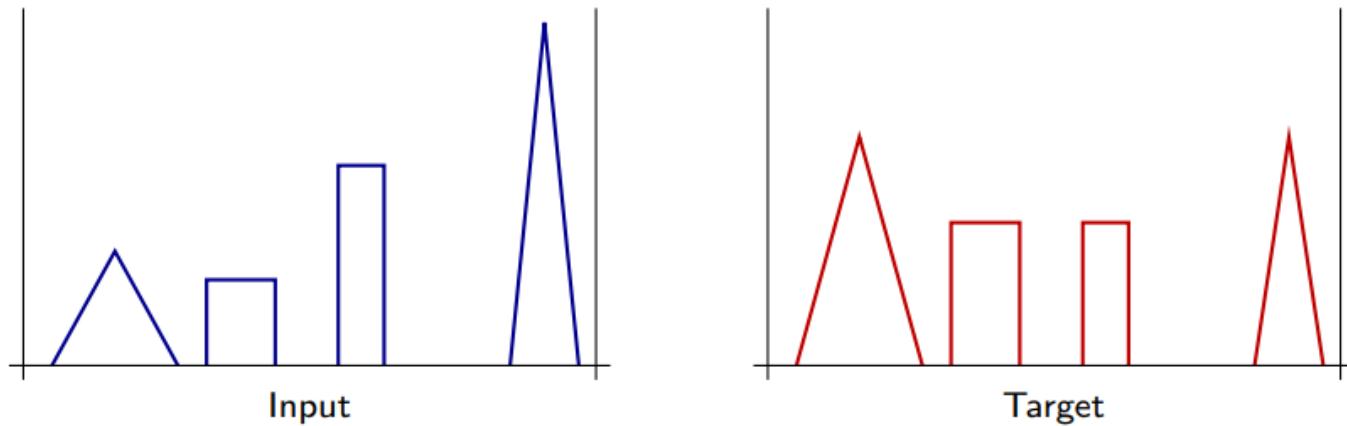


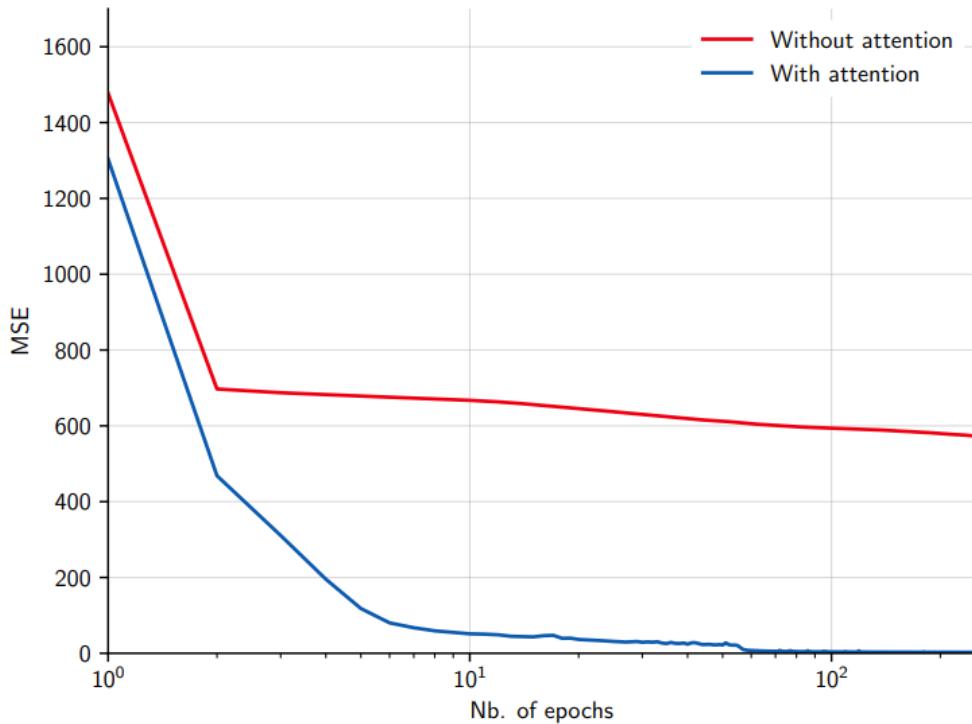
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

де  $n$  – довжина послідовності,  $d$  – розмірність вбудовування, а  $k$  – розмір ядра згортки.

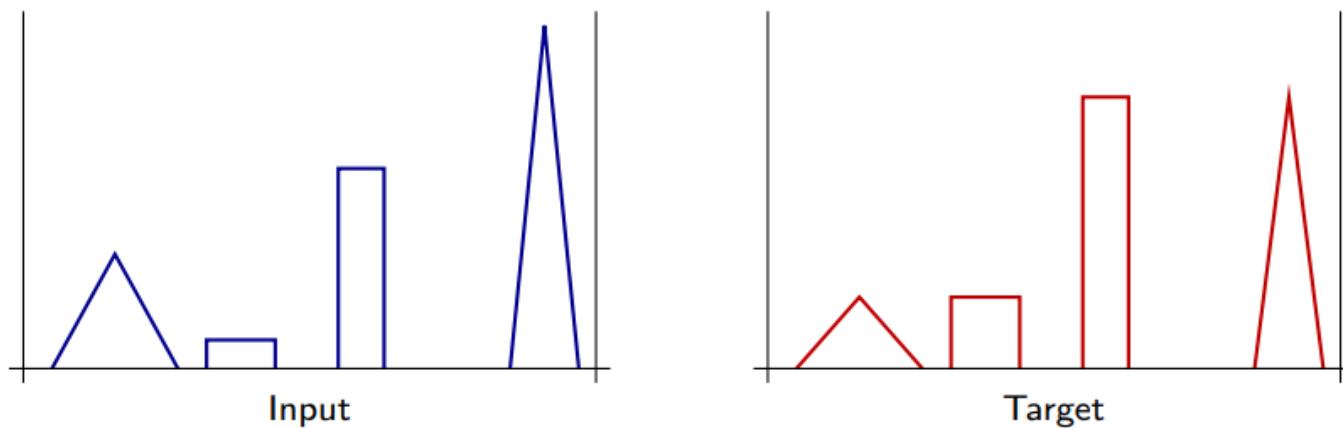
## Приклад

Щоб проілюструвати поведінку механізму уваги, розглянемо просту задачу з 1d послідовностями, що складаються з двох трикутників і двох прямокутників. Цільова послідовність (Target) усереднює висоти в кожній парі фігур.





Ми можемо модифікувати задачу, щоб розглянути мету (Target), де парами для усереднення є дві крайні фігури праворуч і ліворуч.

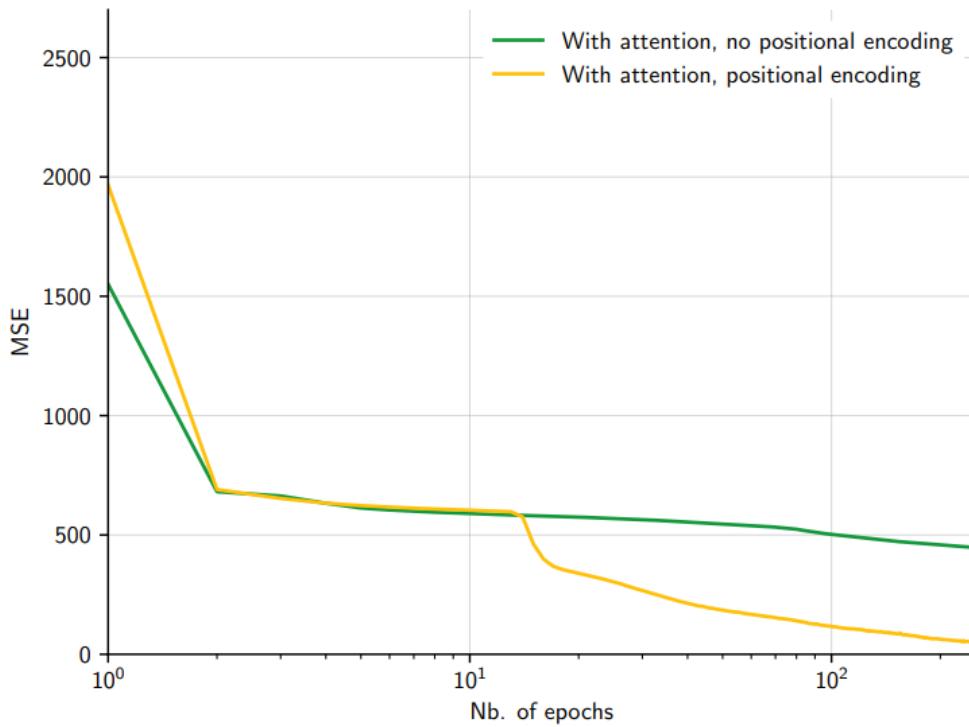


Очікується, що продуктивність буде низькою, враховуючи нездатність шару самоуваги враховувати абсолютні або відносні позиції. Дійсно, самоувага є інваріантною до перестановки:

$$\begin{aligned}\mathbf{y} &= \sum_{i=1}^m \text{softmax}_i \left( \frac{\mathbf{q}^T \mathbf{K}_i^T}{\sqrt{d}} \right) \mathbf{V}_i \\ &= \sum_{i=1}^m \text{softmax}_i \left( \frac{\mathbf{q}^T \mathbf{K}_{\sigma(i)}^T}{\sqrt{d}} \right) \mathbf{V}_{\sigma(i)}\end{aligned}$$

для будь-якої перестановки  $\sigma$  пар ключ-значення.

(Самоувага також еквівалентна перестановці  $\sigma$  запитів.)



Однак цю проблему можна вирішити, надавши позиційне кодування явно шару уваги.

# Трансформери

Vaswani та ін. (2017) запропонували піти ще далі: замість використання механізму уваги як доповнення до стандартних згорткових і рекурентних шарів, вони розробили будівельний блок трансформера, що складається виключно з шарів уваги.

Трансформер був розроблений для завдання перекладу послідовність-послідовність, але в даний час цей блок є ключовим елементом найсучасніших підходів до більшості завдань, що стосуються обробки наборів або послідовностей.

# Attention Is All You Need

**Ashish Vaswani\***  
Google Brain  
[avaswani@google.com](mailto:avaswani@google.com)

**Noam Shazeer\***  
Google Brain  
[noam@google.com](mailto:noam@google.com)

**Niki Parmar\***  
Google Research  
[nikip@google.com](mailto:nikip@google.com)

**Jakob Uszkoreit\***  
Google Research  
[usz@google.com](mailto:usz@google.com)

**Llion Jones\***  
Google Research  
[llion@google.com](mailto:llion@google.com)

**Aidan N. Gomez\*** <sup>†</sup>  
University of Toronto  
[aidan@cs.toronto.edu](mailto:aidan@cs.toronto.edu)

**Lukasz Kaiser\***  
Google Brain  
[lukaszkaiser@google.com](mailto:lukaszkaiser@google.com)

**Illia Polosukhin\*** <sup>‡</sup>  
[illia.polosukhin@gmail.com](mailto:illia.polosukhin@gmail.com)

## Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

\*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

<sup>†</sup>Work performed while at Google Brain.

<sup>‡</sup>Work performed while at Google Research.



# Ілля Полосухін

Національний технічний університет  
«Харківський політехнічний інститут»

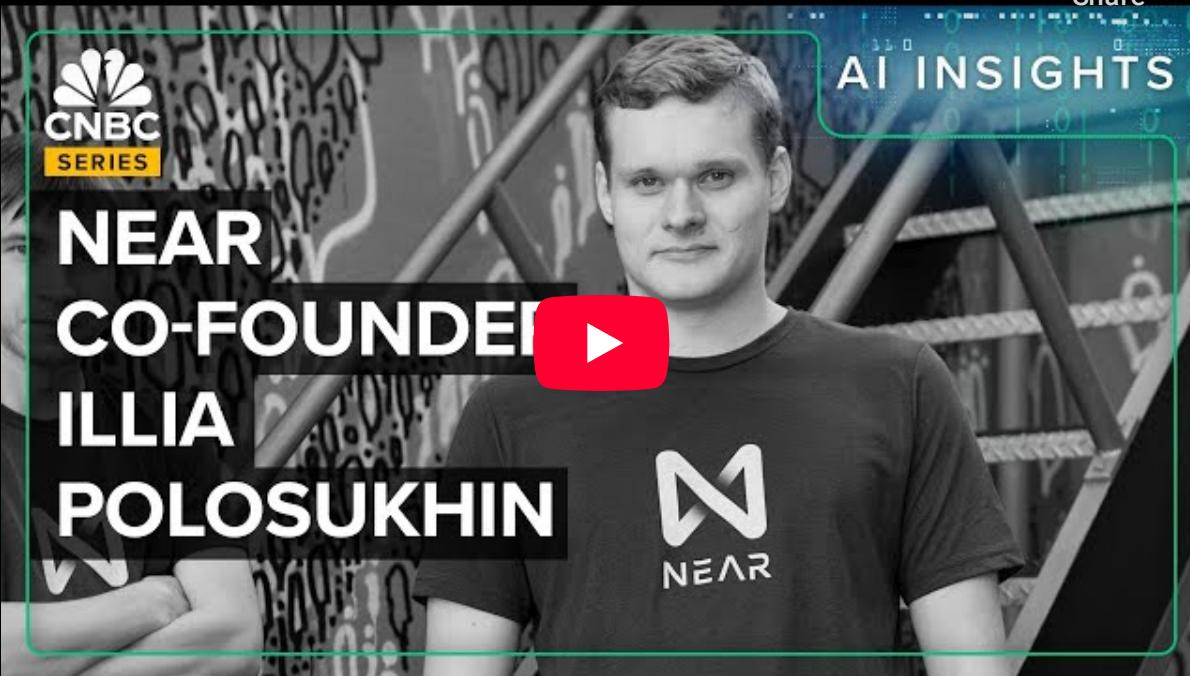
Співавтор наукової роботи “Attention Is All You Need”, яка започаткувала епоху трансформерів у штучному інтелекті.



Illia Polosukhin On Inventing The Tech Behind Generativ...



Share



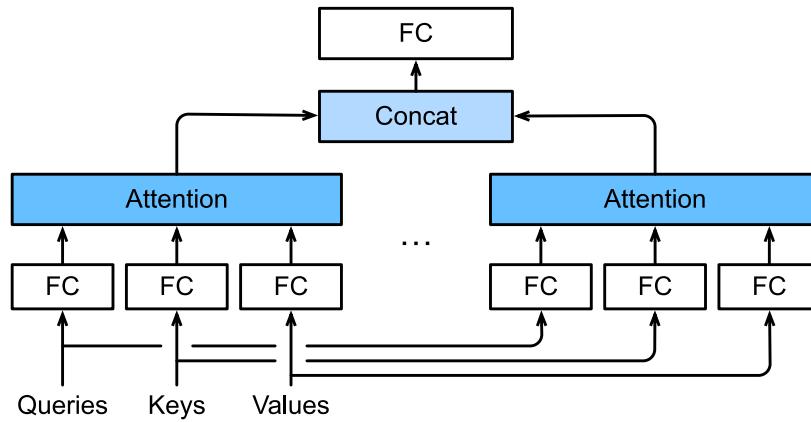
Illia Polosukhin On Inventing The Tech Behind Generative AI At Google (CNBC, 2024)

## Масштабований скалярний добуток уваги

Першим будівельним блоком архітектури трансформера є масштабований блок уваги на основі скалярного добутку:

$$\text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

де масштабування  $1/\sqrt{d_k}$  використовується для підтримки постійної температури (softmax) для різних варіантів вибору розмірності запиту/ключа  $d_k$ .



## Multi-head attention (увага з кількома паралельними блоками)

Трансформер проєктує запити (queries), ключі (keys) і значення (values)  $h = 8$  разів (transformer base model Vaswani et al. (2017)) за допомогою лінійних проєкцій, у розмірності  $d_q = 64, d_k = 64$  і  $d_v = 64$  відповідно.

$$\text{multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat} (\mathbf{H}_1, \dots, \mathbf{H}_h) \mathbf{W}^O$$

$$\mathbf{H}_i = \text{attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

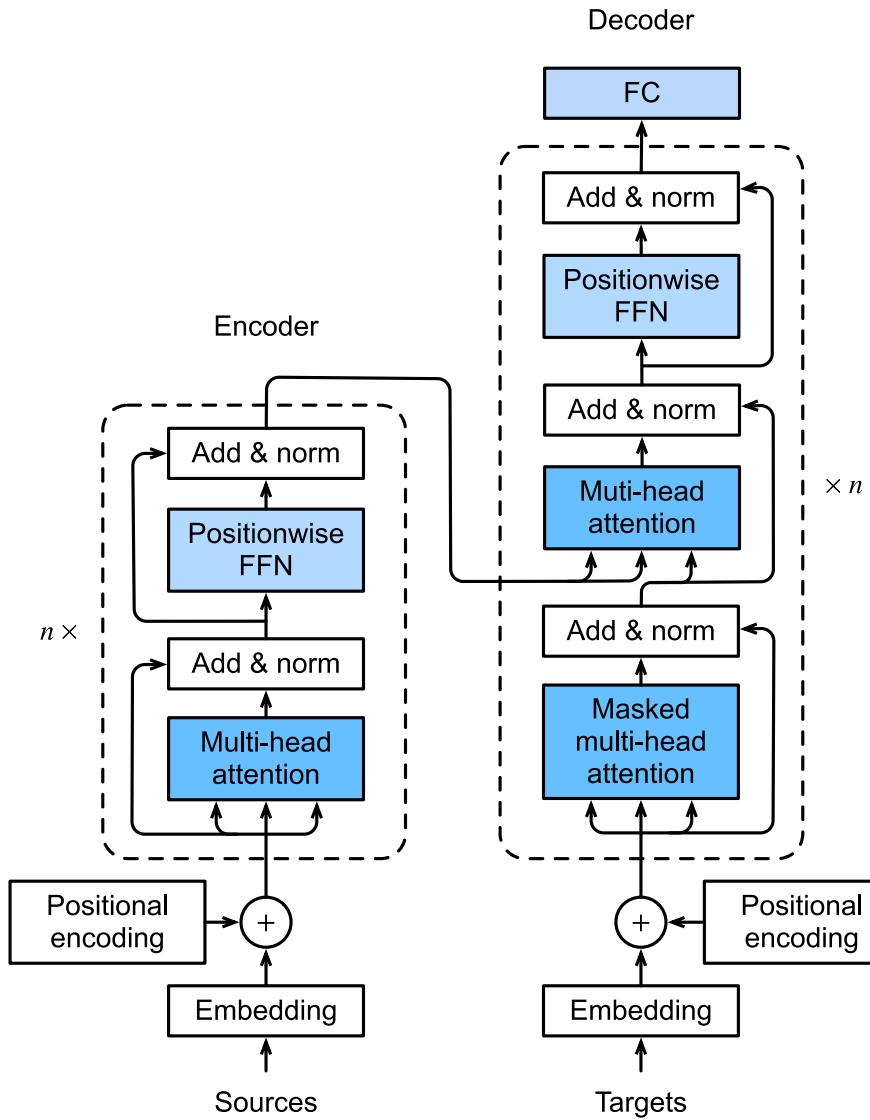
де  $d_{\text{model}} = 512$

$$\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \mathbf{W}_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

## Архітектура кодер-декодер

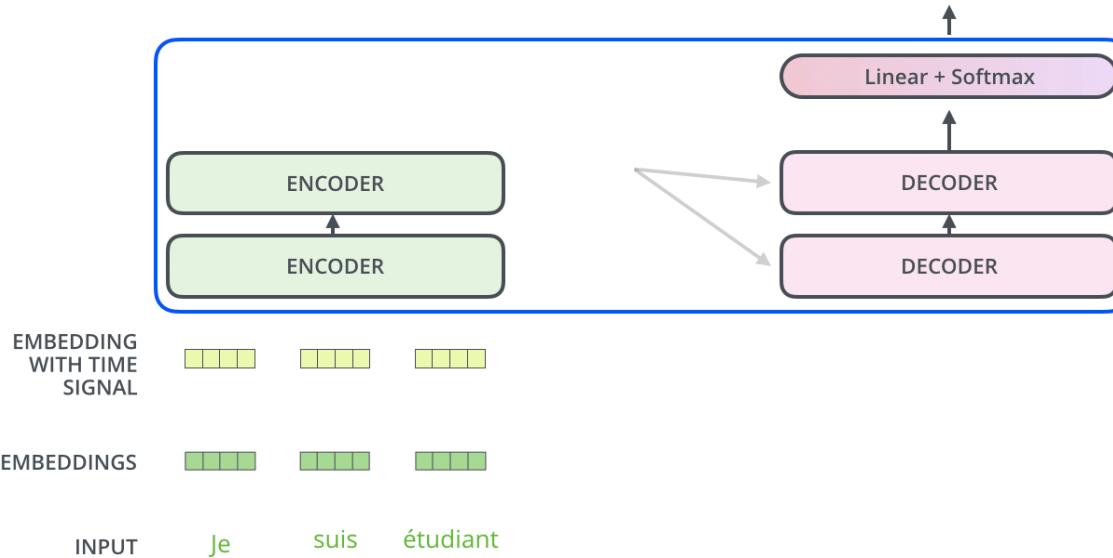
Модель трансформера складається з:

- Кодер, що поєднує  $N = 6$  модулів, кожен з яких складається з підмодулів уваги з кількома паралельними блоками та одношарового MLP (з одним прихованим шаром) для кожного компонента, з залишковим проходженням та нормалізацією шару. Всі підмодулі та шари вбудовування генерують вихідні дані розміром  $d_{\text{model}} = 512$ .
- Декодер, що поєднує  $N = 6$  модулів, подібних до кодера, але використовує приховану самоувагу, щоб запобігти врахування наступних позицій при обчисленні поточної. Крім того, додає третій підмодуль, який виконує **multi-head attention** над виходом стека кодера.



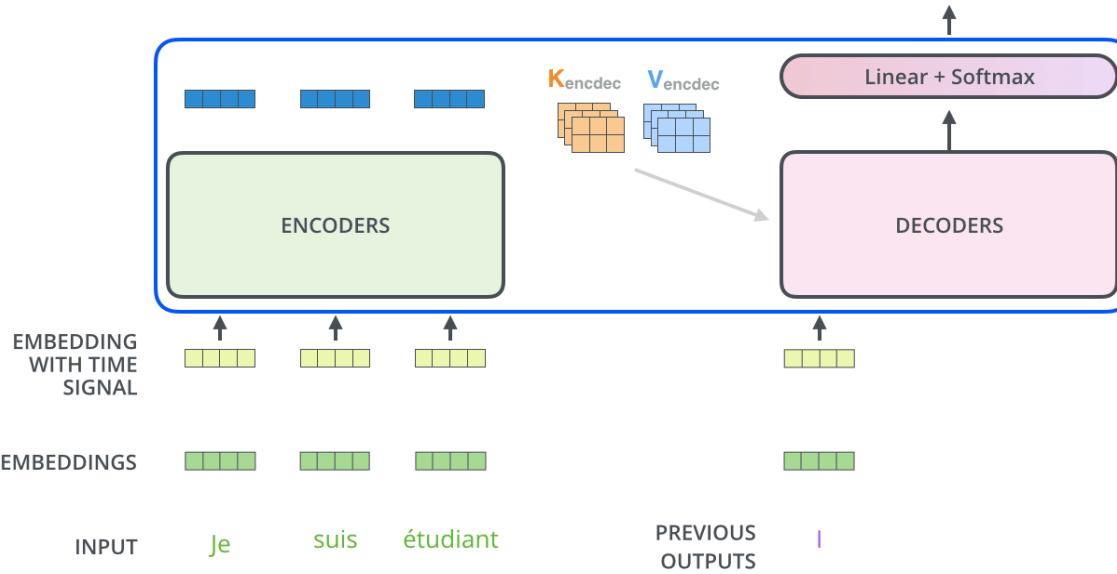
Decoding time step: 1 2 3 4 5 6

OUTPUT



Кодери починають з обробки вхідної послідовності. Вихідні дані верхнього кодера потім перетворюються в набір векторів уваги **K** і **V**, які передаються декодерам.

Decoding time step: 1 2 3 4 5 6      OUTPUT |



Кожен крок у фазі декодування генерує вихідний токен, доки не буде досягнуто спеціального символу, що вказує на завершення виходу декодера трансформера.

Вихід кожного кроку подається на нижній декодер на наступному часовому кроці, а декодери «піднімають» свої результати декодування вгору точно так само, як це робили енкодери.

У декодері:

- Перший підмодуль (masked self-attention) може враховувати лише попередні позиції у вихідній послідовності.
- Другий підмодуль **multi-head attention** працює так само, як **multi-head self-attention**, за винятком того, що матрицю запитів (Q) він формує з нижнього шару, а матриці ключів (K) і значень (V) беруться з виходу стеку кодера.

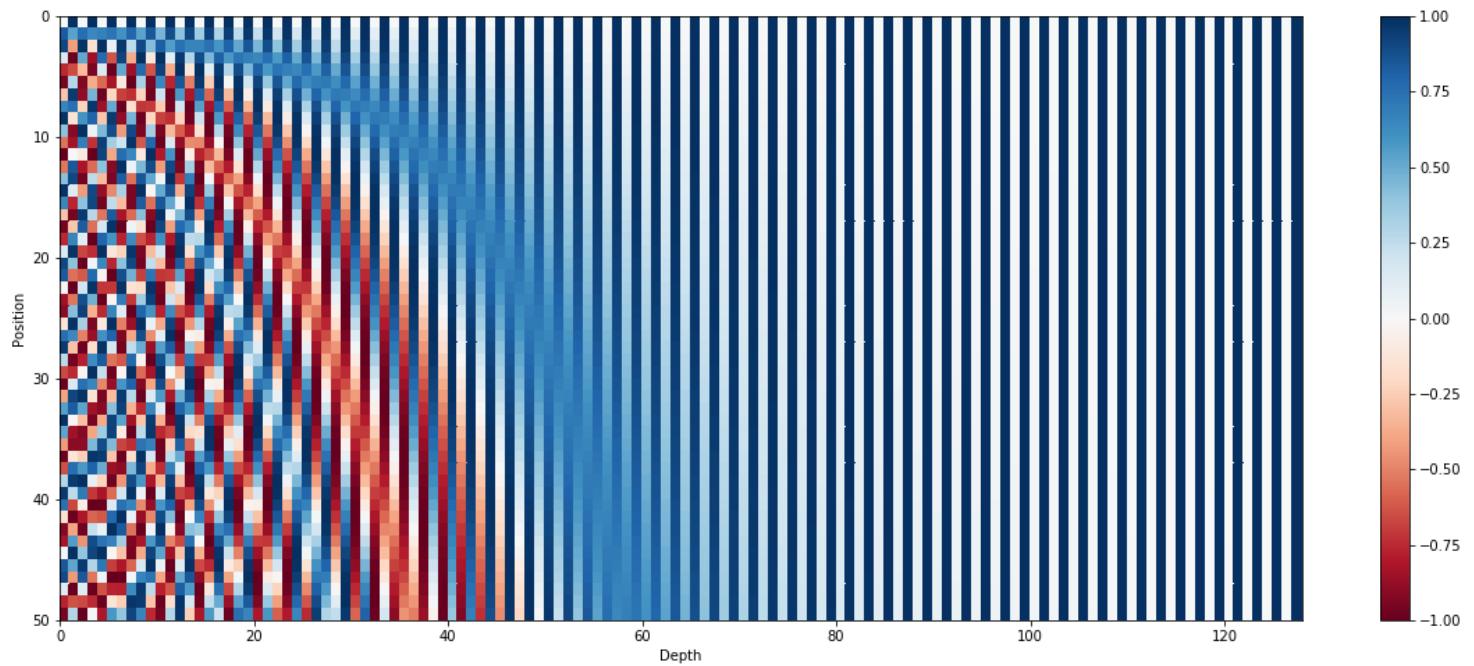
## Positional encoding (позиційне кодування)

Інформація про положення надається за допомогою **адитивного** кодування положення того ж розміру  $d_{\text{model}}$ , що і внутрішнє представлення, і має вигляд:

$$\begin{aligned} \text{PE}_{t,2i} &= \sin \left( \frac{t}{10000^{\frac{2i}{d_{\text{model}}}}} \right) \\ \text{PE}_{t,2i+1} &= \cos \left( \frac{t}{10000^{\frac{2i}{d_{\text{model}}}}} \right). \end{aligned}$$

Після додавання позиційного кодування слова будуть розташовуватися біжче одне до одного залежно від схожості їх значення та відносного положення в реченні у  $d_{\text{model}}$ -вимірному просторі.

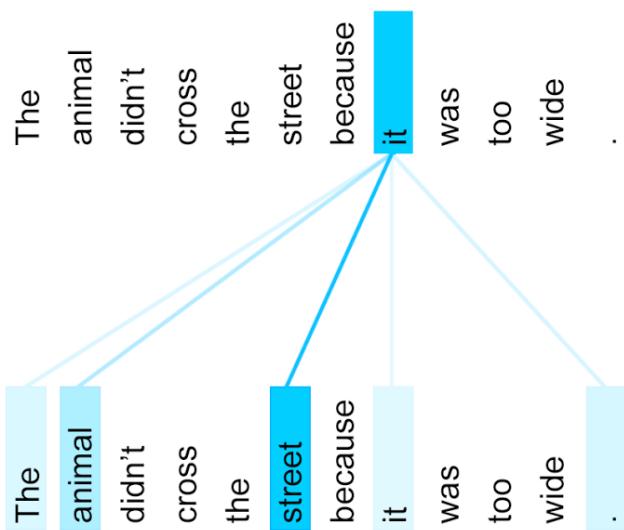
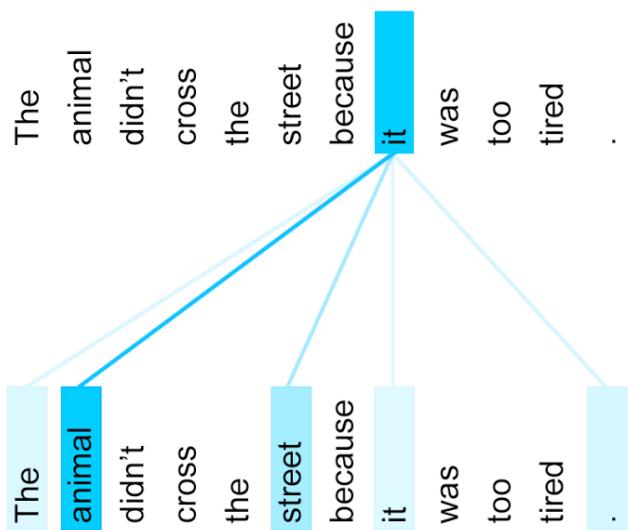
Альтернативно, модель може також навчитися позиційному кодуванню.



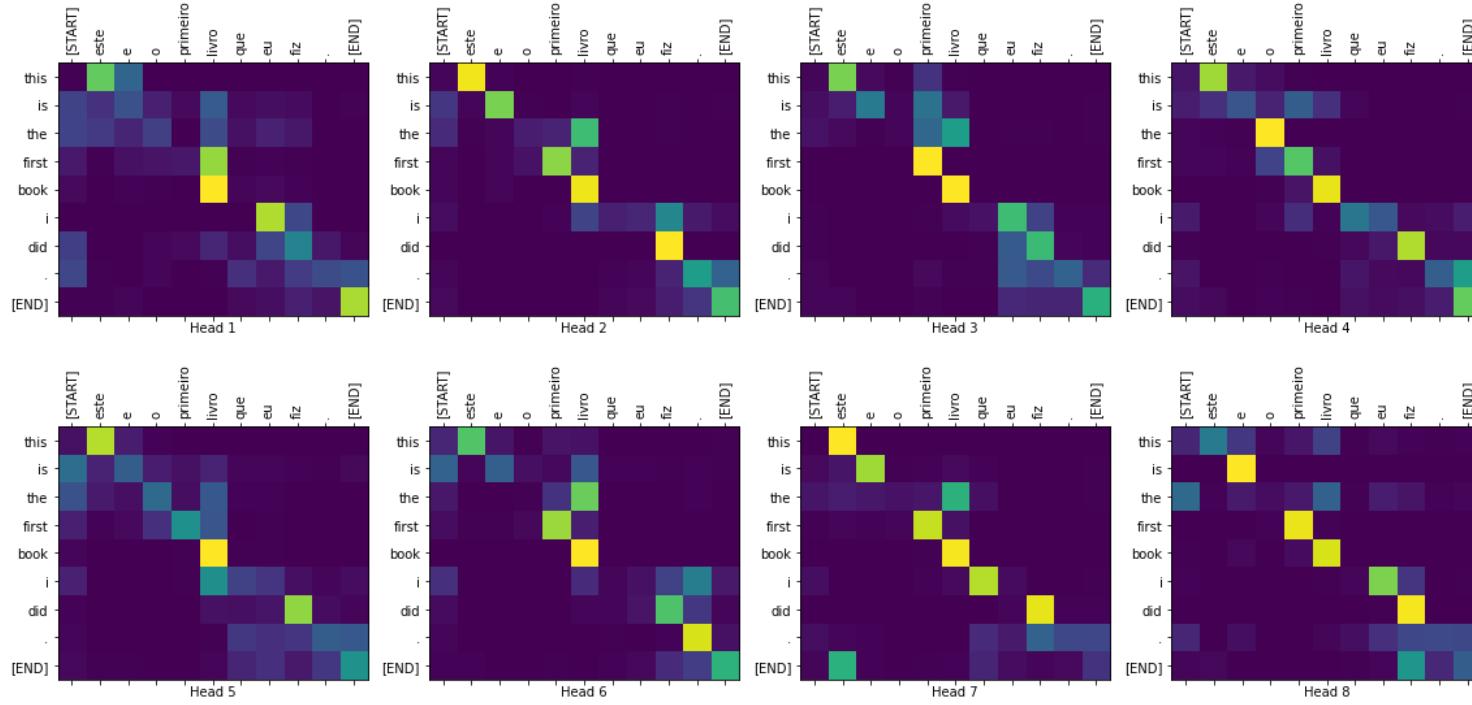
128-вимірне позиційне кодування для речення з максимальною довжиною 50 слів. Кожен рядок представляє вектор ембеддингу для відповідної позиції.

## Машинний переклад

Архітектура трансформера вперше була розроблена для машинного перекладу та протестована на завданнях англійська → німецька і англійська → французька.



Шари самоуваги (self-attention) навчилися розуміти, що слово «it» може відноситися до різних сутностей у різних контекстах.

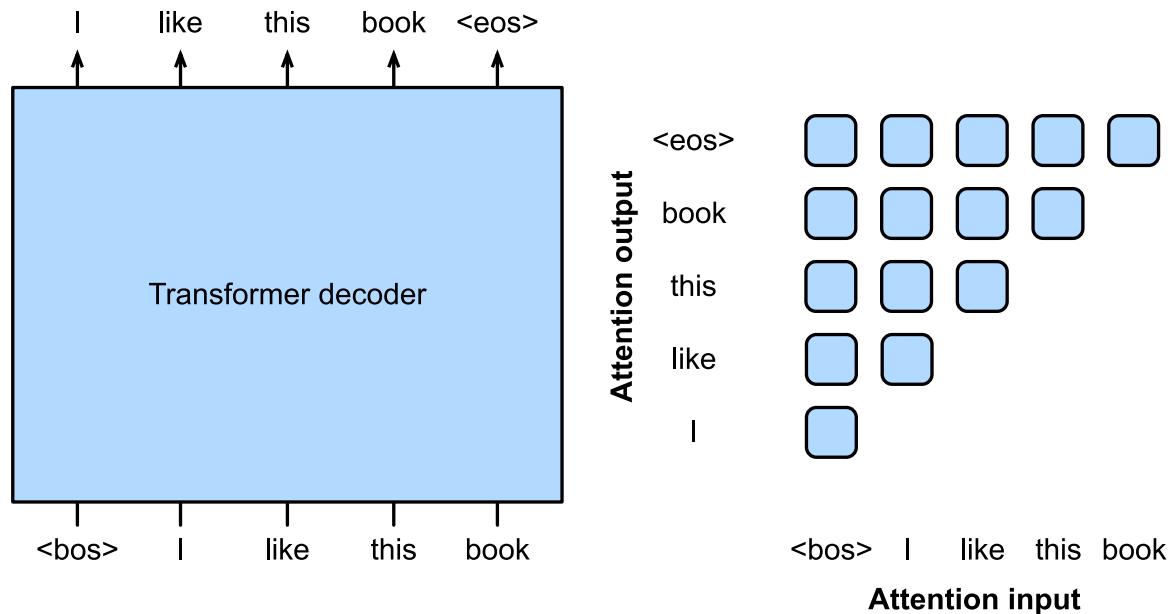


Карти уваги (attention maps), отримані з модулів multi-head attention, показують, як вхідні токени пов'язані з вихідними токенами.

## Трансформери тільки з декодером

Декодерний трансформер став фактичною стандартною архітектурою для великих мовних моделей (LLM):  $p(\mathbf{x}_t | \mathbf{x}_{1:t-1})$ .

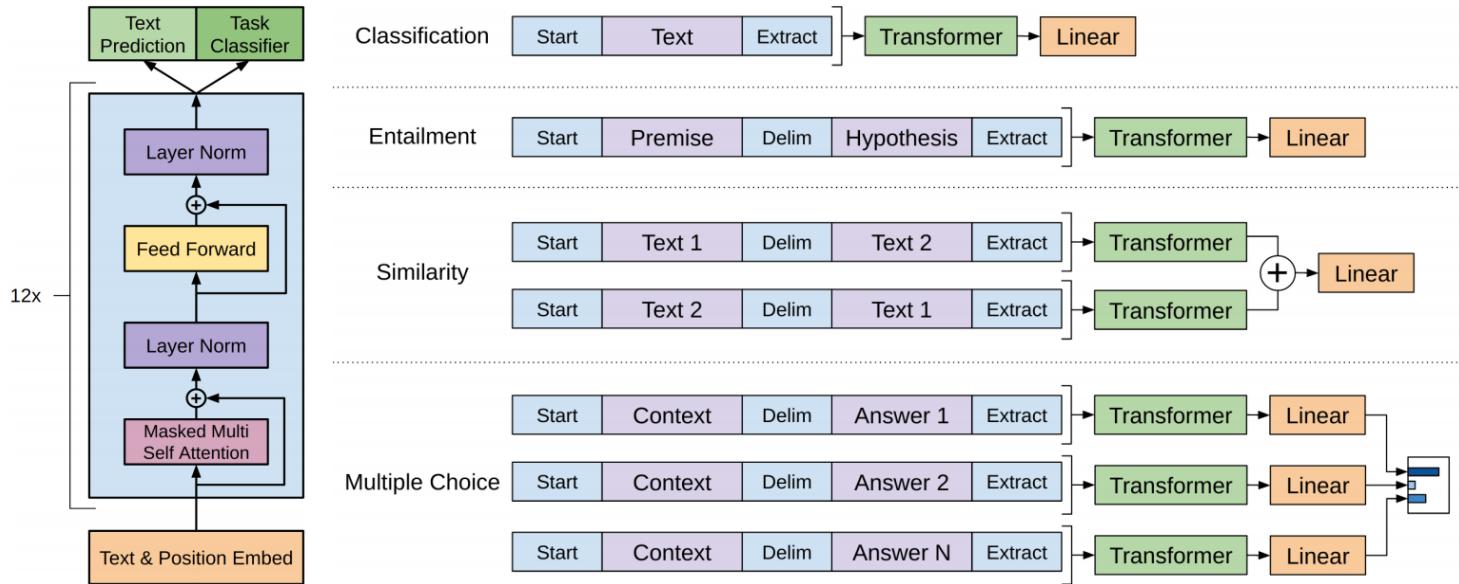
Ці моделі навчаються за схемою self-supervised learning (само-контрольованого навчання), де цільова послідовність збігається з вхідною, але зсунута на один токен вправо.





Демо

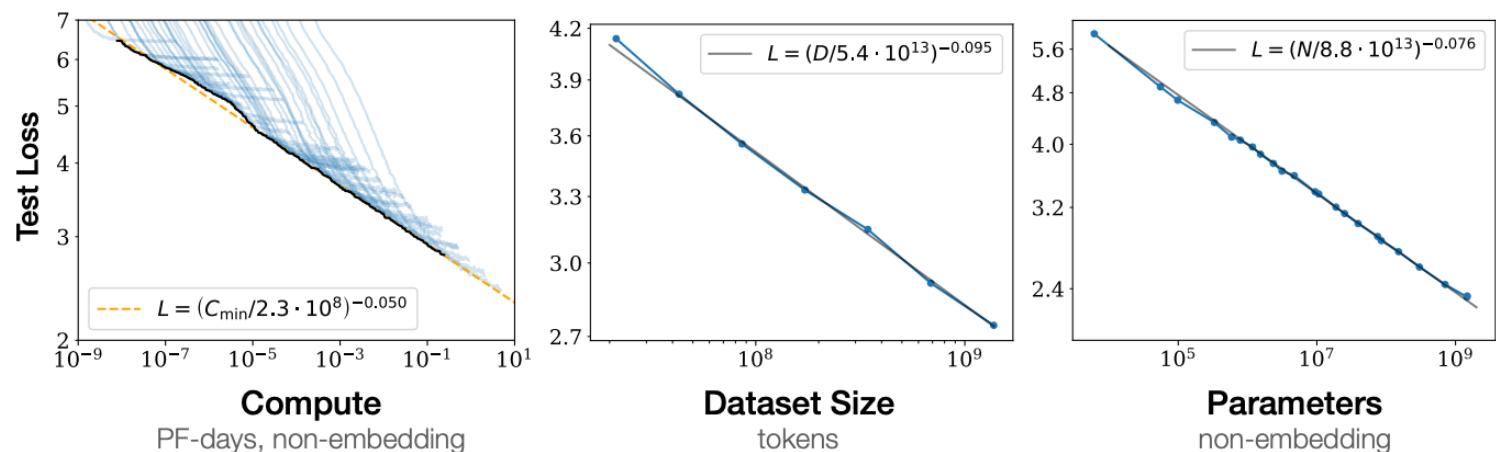
Історично, GPT-1 спочатку пройшов попереднє навчання, а потім був донавчений (fine-tuned) на конкретних завданнях.



## Закони масштабування

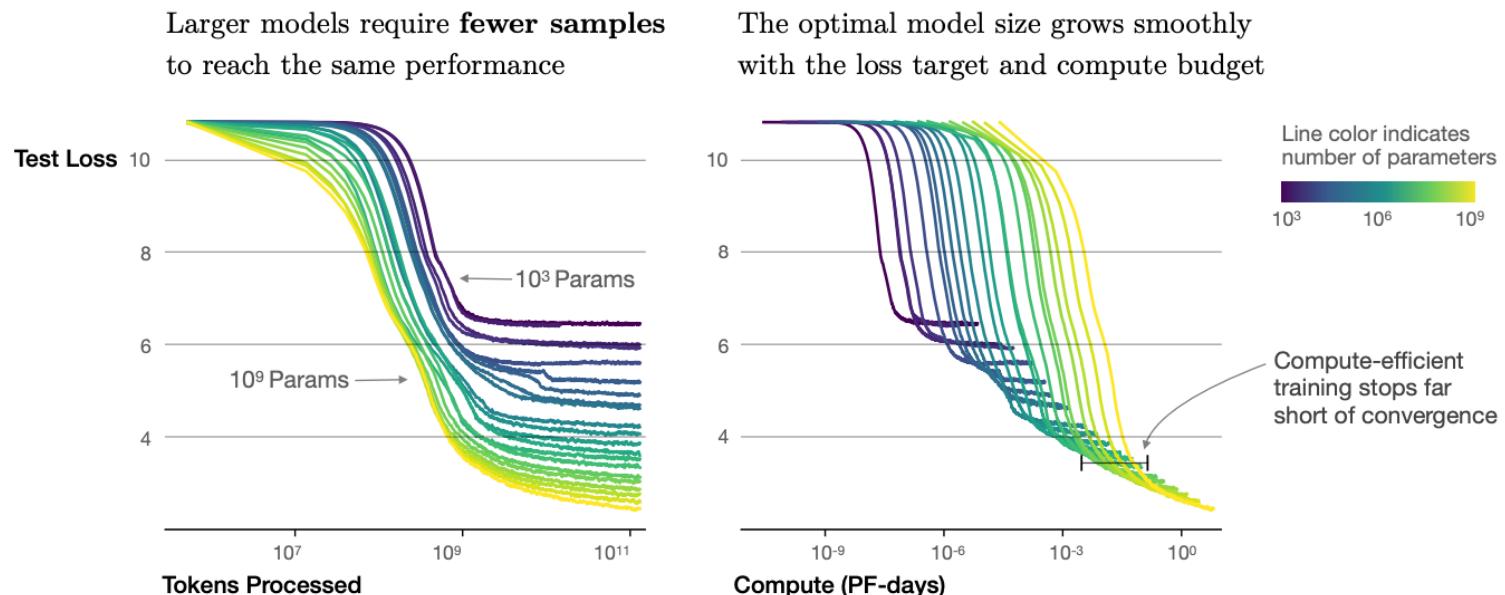
Продуктивність мовної моделі на основі трансформера покращується поступово зі збільшенням розміру моделі (кількість параметрів), розміру набору даних, обсягу обчислювальних ресурсів, використаних для навчання.

Для досягнення оптимальної продуктивності всі три фактори: розмір моделі, обсяг даних та обчислювальні ресурси повинні збільшуватися одночасно. Емпірично встановлено, що продуктивність моделі залежить від кожного фактора за степеневим законом.



Великі моделі також демонструють кращу ефективність використання прикладів порівняно з малими моделями.

- Більші моделі потребують менше даних, щоб досягти тієї ж продуктивності.
- Оптимальний розмір моделі зростає поступово, залежно від обсягу обчислень, доступних для навчання.



# Агенти для спілкування

## ChatGPT

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

[ChatGPT Feb 13 Version](#). Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

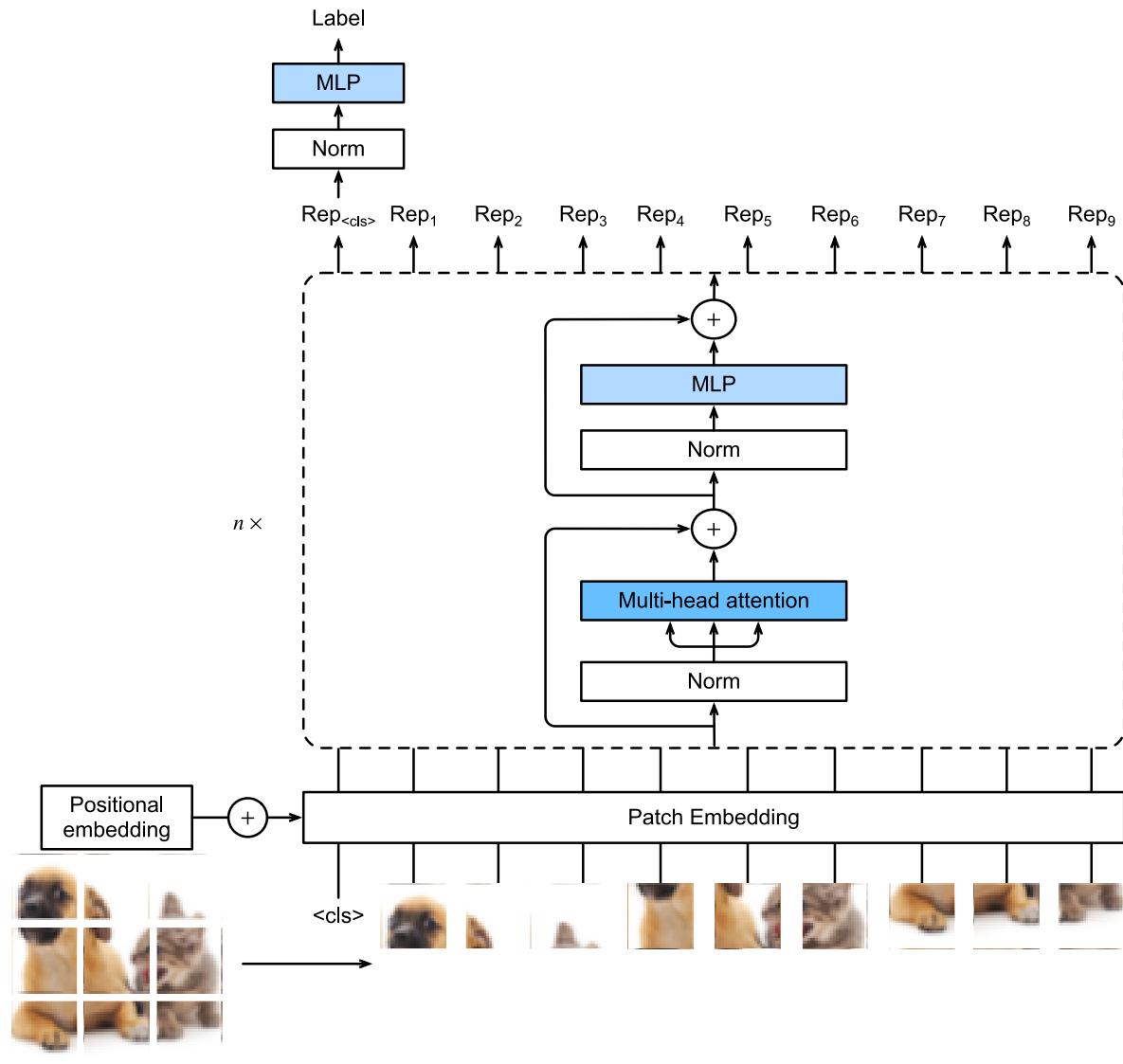
Всі сучасні розмовні агенти базуються на трансформерних моделях, масштабованих до мільярдів параметрів, трильйонів токенів для навчання та тисяч PF-day обчислень.

- PF-day = 1 пета FLOPS протягом одного дня.

# Трансформеры для зображень

Архітектура трансформера вперше була розроблена для послідовностей, але її можна адаптувати для обробки зображень.

Ключова ідея полягає в тому, щоб перетворити вхідне зображення на послідовність патчів, які потім обробляються кодером трансформера. Ця архітектура відома як зоровий трансформер (**Vision Transformer - ViT**).



- Вхідне зображення ділиться на неперетинаючі патчі, які потім лінійно перетворюються в послідовність векторів (embeddings).
- Ця послідовність векторів обробляється кодером трансформера, який видає нову послідовність векторів на виході.
- Навчання ViT може проводитися як (supervised), так і (self-supervised).

