За умови належного зазначення авторства Google цим дозволяє відтворювати таблиці та рисунки цієї статті виключно для використання в журналістських або наукових працях.

Увага — це все, що вам потрібно

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
Університет Торонто
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Анотація

Провідні моделі перетворення послідовностей базуються на складних рекурентних або згорткових нейронних мережах, що містять кодувальник і декодувальник (encoder-decoder). Найкращі моделі також з'єднують кодувальник із декодувальником механізмом уваги (attention). Ми пропонуємо нову просту архітектуру мережі, Transformer, що базується виняткова на механізмах уваги, повністю відмовляючись від рекурентності та згорток. Експерименти на двох задачах машинного перекладу показують, що ці моделі перевершують інші за якістю, краще піддаються розпаралелюванню та потребують істотно менше часу на навчання. Наша модель досягає 28.4 BLEU в задачі англо-німецького перекладу WMT 2014, поліпшуючи наявні найкращі результати, включно з ансамблями, більш ніж на 2 BLEU. У задачі англо-французького перекладу WMT 2014 наша модель встановлює новий найкращий результат одиночної моделі — 41.8 BLEU після 3.5 дня навчання на восьми графічних процесорах, що становить лише малу частину витрат на навчання найкращих моделей з літератури. Ми показуємо, що Transformer добре узагальнює на інші задачі, успішно застосовуючи його до синтаксичного аналізу англійської за складниками як із великими, так і з обмеженими навчальними даними.

[*]Рівний внесок. Порядок авторів випадковий. Jakob запропонував замінити PHM самоувагою та започаткував перевірку цієї ідеї. Ashish разом з Illia спроєктував і реалізував перші моделі Transformer та відігравав ключову роль у всіх аспектах цієї роботи. Noam запропонував масштабовану скалярнодобуткову увагу, багатоголову увагу та безпараметричне представлення позицій і став ще одним учасником, залученим майже до кожної деталі. Niki спроєктувала, реалізувала, налаштувала й оцінила безліч варіантів моделі в нашій початковій кодовій базі та tensor2tensor. Llion також експериментував із новими варіантами моделі, відповідав за початкову кодову базу, ефективне виведення та візуалізації. Lukasz та Aidan провели незліченні довгі дні, проєктуючи різні частини й реалізуючи tensor2tensor, який замінив попередню кодову базу, значно поліпшив результати та різко прискорив наші дослідження.

[†]Роботу виконано під час перебування в Google Brain.

[‡]Роботу виконано під час перебування в Google Research.

31-ша конференція з нейронних систем обробки інформації (NIPS 2017), Long Beach, CA, USA.

1 Вступ

Рекурентні нейронні мережі, зокрема мережі довгої короткочасної пам'яті [13] та вентильні рекурентні нейронні мережі [7], міцно утвердилися як найкращі підходи до моделювання та перетворення послідовностей, зокрема мовного моделювання й машинного перекладу [35, 2, 5]. Відтоді численні дослідження продовжують розширювати можливості рекурентних мовних моделей та архітектур кодувальника-декодувальника [38, 24, 15].

Рекурентні моделі зазвичай розкладають обчислення за позиціями символів у вхідних і вихідних послідовностях. Зіставляючи позиції з кроками обчислювального часу, вони генерують послідовність прихованих станів h_t як функцію попереднього прихованого стану h_{t-1} та входу в позиції t. Така послідовна за своєю природою організація унеможливлює розпаралелювання всередині навчальних прикладів. Це стає критичним для довших послідовностей, оскільки обмеження пам'яті звужують можливості об'єднання прикладів у пакети. Нещодавні роботи істотно підвищили обчислювальну ефективність завдяки прийомам факторизації [21] та умовним обчисленням [32]; останні також поліпшили якість моделей. Проте фундаментальне обмеження послідовних обчислень зберігається.

Механізми уваги стали невід'ємною складовою успішних моделей моделювання та перетворення послідовностей у різних задачах, даючи змогу моделювати залежності незалежно від відстані між ними у вхідних чи вихідних послідовностях [2, 19]. Однак майже завжди, за кількома винятками [27], такі механізми уваги використовують разом із рекурентною мережею.

У цій роботі ми пропонуємо Transformer: архітектуру моделі, що відмовляється від рекурентності й повністю спирається на механізм уваги для встановлення глобальних залежностей між входом і виходом. Transformer допускає значно більше розпаралелювання та досягає нового найкращого рівня якості перекладу лише після дванадцяти годин навчання на восьми графічних процесорах P100.

2 Передумови

Мета скоротити послідовні обчислення лежить також в основі Extended Neural GPU [16], ByteNet [18] та ConvS2S [9]. Усі вони використовують згорткові нейронні мережі як базовий будівельний блок, паралельно обчислюючи приховані представлення для всіх вхідних і вихідних позицій. У цих моделях кількість операцій, необхідних для зв'язування сигналів із двох довільних вхідних чи вихідних позицій, зростає зі збільшенням відстані між позиціями: лінійно для ConvS2S та логарифмічно для ByteNet. Це ускладнює навчання залежностей між віддаленими позиціями [12]. У Transformer ця кількість стає сталою, хоча й ціною зменшення ефективної роздільності через усереднення позицій, зважених увагою. Цьому ефекту ми протидіємо багатоголовою увагою, описаною в розділі 3.2.

Самоувага (self-attention), яку іноді називають внутрішньою увагою, — це механізм уваги, що пов'язує різні позиції однієї послідовності для обчислення її представлення. Самоувагу успішно застосовували в різноманітних задачах, зокрема розумінні прочитаного, абстрактивному підсумовуванні, визначенні логічного випливання в тексті та навчанні незалежних від задачі представлень речень [4, 27, 28, 22].

Наскрізні мережі пам'яті базуються на рекурентному механізмі уваги замість рекурентності, прив'язаної до послідовності, та продемонстрували добрі результати в задачах відповідей на запитання простою мовою і мовного моделювання [34].

Проте, наскільки нам відомо, Transformer є першою моделлю перетворення послідовностей, яка повністю покладається на самоувагу для обчислення представлень входу й виходу, не використовуючи прив'язаних до послідовності РНМ або згорток. У наступних розділах ми опишемо Transformer, обґрунтуємо самоувагу та обговоримо її переваги порівняно з моделями на зразок [17, 18] та [9].

3 Архітектура моделі

Більшість конкурентоспроможних нейронних моделей перетворення послідовностей мають структуру кодувальника-декодувальника [5, 2, 35]. Кодувальник відображає вхідну послідовність представлень символів $(x_1, ..., x_n)$ у послідовність неперервних представлень $\mathbf{z} = (z_1, ..., z_n)$. Маючи $\mathbf{z}$, декодувальник генерує вихідну послідовність символів $(y_1, ..., y_m)$ по одному елементу. На кожно-

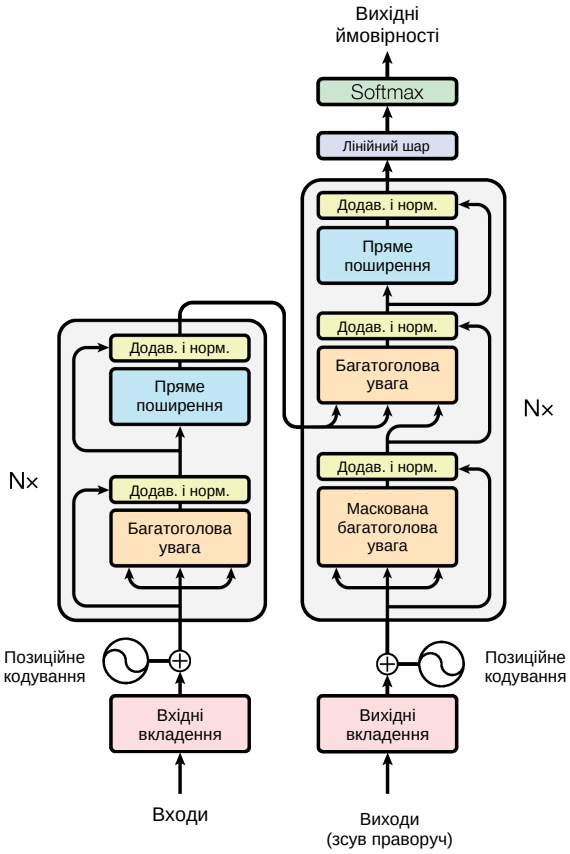


Рисунок 1: Transformer — архітектура моделі.

му кроці модель є авторегресивною [10]: генеруючи наступний символ, вона використовує раніше згенеровані символи як додатковий вхід.

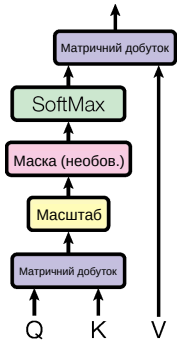
Transformer дотримується цієї загальної архітектури, використовуючи стеки шарів самоуваги та поточкових повнозв'язних шарів у кодувальнику й декодувальнику, показаних відповідно в лівій і правій половинах рис. 1.

3.1 Стеки кодувальника та декодувальника

Кодувальник: Кодувальник складається зі стека $N = 6$ однакових шарів. Кожен шар має два підшари. Перший — механізм багатоголової уваги (multi-head attention), застосований як самоувага, а другий — проста позиційна повнозв'язна мережа прямого поширення (feed-forward network). Навколо кожного з двох підшарів ми використовуємо залишкове з'єднання (residual connection) [11], за яким іде нормалізація шару (layer normalization) [1]. Тобто вихід кожного підшару дорівнює $\text{LayerNorm}(x + \text{Sublayer}(x))$, де $\text{Sublayer}(x)$ — функція, реалізована самим підшаром. Для забезпечення цих залишкових з'єднань усі підшари моделі, а також шари векторних вкладень, формують виходи розмірності $d_{\text{model}} = 512$.

Декодувальник: Декодувальник також складається зі стека $N = 6$ однакових шарів. Окрім двох підшарів, наявних у кожному шарі кодувальника, декодувальник додає третій підшар, який виконує багатоголову увагу над виходом стека кодувальника. Як і в кодувальнику, навколо кожного підшару ми використовуємо залишкові з'єднання з подальшою нормалізацією шару. Ми також змінюємо підшар самоуваги в стеку декодувальника, щоб позиції не могли звертати увагу на наступні позиції. Це маскування разом зі зсувом вихідних вкладень на одну позицію гарантує, що передбачення для позиції i залежать лише від відомих виходів у позиціях, менших за i.

Масштабована скалярнодобуткова увага



Багатоголова увага

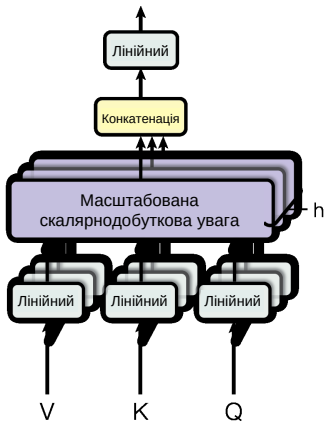


Рисунок 2: Ліворуч: масштабована скалярнодобуткова увага. Праворуч: багатоголова увага складається з кількох шарів уваги, які працюють паралельно.

3.2 Увага

Функцію уваги можна описати як відображення запиту та набору пар ключ-значення у вихід, причому запит, ключі, значення та вихід є векторами. Вихід обчислюють як зважену суму значень, де вагу кожного значення визначає функція сумісності запиту з відповідним ключем.

3.2.1 Масштабована скалярнодобуткова увага

Наш конкретний різновид уваги ми називаємо «масштабованою скалярнодобутковою увагою» (scaled dot-product attention; рис. 2). Вхід складається із запитів та ключів розмірності d_k і значень розмірності d_v. Ми обчислюємо скалярні добутки запиту з усіма ключами, ділимо кожен на $\sqrt{d_k}$ і застосовуємо функцію softmax, щоб отримати ваги значень.

На практиці функцію уваги обчислюємо одночасно для набору запитів, об'єднаних у матрицю Q. Ключі та значення також об'єднуємо в матриці K і V. Матрицю виходів обчислюємо так:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

Дві найпоширеніші функції уваги — адитивна увага [2] та скалярнодобуткова (мультиплікативна) увага. Скалярнодобуткова увага ідентична нашому алгоритму, окрім масштабувального множника $\frac{1}{\sqrt{d_k}}$. Адитивна увага обчислює функцію сумісності за допомогою мережі прямого поширення з одним прихованим шаром. Хоча теоретична складність обох підходів подібна, на практиці скалярнодобуткова увага значно швидша й економніша за пам'яттю, оскільки її можна реалізувати високооптимізованим кодом множення матриць.

За малих значень d_k обидва механізми працюють подібно, проте за більших значень d_k адитивна увага перевершує скалярнодобуткову без масштабування [3]. Ми припускаємо, що за великих d_k скалярні добутки стають великими за модулем, переводячи функцію softmax в області з надзвичайно малими градієнтами[4]. Щоб протидіяти цьому ефекту, ми масштабуємо скалярні добутки множником $\frac{1}{\sqrt{d_k}}$.

3.2.2 Багатоголова увага

Замість одноразового виконання функції уваги з ключами, значеннями та запитами розмірності d_{model} виявилося корисним h разів лінійно проєктувати запити, ключі та значення різними навченими лінійними проєкціями до розмірностей d_k, d_k та d_v відповідно. Для кожного з цих проєктованих варіантів запитів, ключів та значень ми паралельно виконуємо функцію уваги, отримуючи вихідні значення розмірності d_v. Їх конкатенують та ще раз проєктують, отримуючи остаточні значення, як показано на рис. 2.

Багатоголова увага дає моделі змогу одночасно звертати увагу на інформацію з різних підпросторів представлень у різних позиціях. За однієї голови уваги усереднення цьому перешкоджає.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{де head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Проєкції задаються матрицями параметрів $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ та $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

У цій роботі ми використовуємо $h = 8$ паралельних шарів уваги, або голів. Для кожної з них $d_k = d_v = d_{\text{model}}/h = 64$. Завдяки меншій розмірності кожної голови загальні обчислювальні витрати подібні до витрат одноголової уваги з повною розмірністю.

3.2.3 Застосування уваги в нашій моделі

Transformer використовує багатоголову увагу трьома способами:

- У шарах «уваги кодувальника-декодувальника» запити надходять із попереднього шару декодувальника, а ключі й значення пам'яті — з виходу кодувальника. Це дає кожній позиції декодувальника змогу звертати увагу на всі позиції вхідної послідовності. Такий підхід відтворює типові механізми уваги кодувальника-декодувальника в моделях перетворення послідовностей на зразок [38, 2, 9].
- Кодувальник містить шари самоуваги. У шарі самоуваги всі ключі, значення та запити надходять з одного місця, у цьому випадку — з виходу попереднього шару кодувальника. Кожна позиція кодувальника може звертати увагу на всі позиції попереднього шару кодувальника.
- Подібним чином шари самоуваги в декодувальнику дають кожній його позиції змогу звертати увагу на всі позиції декодувальника до цієї позиції включно. Щоб зберегти авторегресивність, потрібно запобігти потоку інформації ліворуч у декодувальнику. Ми реалізуємо це всередині масштабованої скалярнодобуткової уваги, маскуючи (встановлюючи в $-\infty$) усі значення на вході softmax, що відповідають недопустимим з'єднанням. Див. рис. 2.

3.3 Позиційні мережі прямого поширення

Окрім підшарів уваги, кожен шар нашого кодувальника та декодувальника містить повнозв'язну мережу прямого поширення, яка застосовується до кожної позиції окремо й однаково. Вона складається з двох лінійних перетворень з активацією ReLU між ними.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

Хоча лінійні перетворення однакові для різних позицій, їхні параметри відрізняються від шару до шару. Інший спосіб описати це — дві згортки з розміром ядра 1. Розмірність входу й виходу дорівнює $d_{\text{model}} = 512$, а внутрішнього шару — $d_{ff} = 2048$.

[4]Щоб пояснити, чому скалярні добутки зростають, припустімо, що компоненти q та k — незалежні випадкові величини із середнім 0 і дисперсією 1. Тоді їхній скалярний добуток $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ має середнє 0 та дисперсію d_k.

3.4 Вкладення та softmax

Подібно до інших моделей перетворення послідовностей, ми використовуємо навчені векторні вкладення, щоб перетворювати вхідні й вихідні токени на вектори розмірності d_{model}. Також використовуємо звичайне навчене лінійне перетворення та функцію softmax для перетворення виходу декодувальника на передбачені ймовірності наступного токена. У нашій моделі обидва шари вкладень і лінійне перетворення перед softmax спільно використовують одну матрицю ваг, подібно до [30]. У шарах вкладень ці ваги множимо на $\sqrt{d_{\text{model}}}$.

3.5 Позиційне кодування

Оскільки наша модель не містить ані рекурентності, ані згорток, для використання порядку послідовності ми маємо додати інформацію про відносну чи абсолютну позицію токенів у ній. Для цього до вхідних вкладень унизу стеків кодувальника й декодувальника додаємо позиційні кодування (positional encoding). Вони мають ту саму розмірність d_{model}, що й вкладення, тому їх можна підсумовувати. Існує багато варіантів позиційного кодування, як навчених, так і фіксованих [9].

У цій роботі ми використовуємо синусні й косинусні функції різних частот:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

де pos — позиція, а i — вимір. Отже, кожному виміру позиційного кодування відповідає синусоїда. Довжини хвиль утворюють геометричну прогресію від 2π до $10000 \cdot 2\pi$. Ми обрали цю функцію, припускаючи, що вона дасть моделі змогу легко навчитися звертати увагу за відносними позиціями, оскільки для будь-якого фіксованого зсуву k величину PE_{pos+k} можна подати як лінійну функцію PE_{pos}.

Ми також експериментували з навченими позиційними вкладеннями [9] і виявили, що обидва варіанти дають майже однакові результати (див. рядок (E) табл. 3). Ми обрали синусоїдний варіант, оскільки він може дати моделі змогу екстраполювати на послідовності, довші за ті, які траплялися під час навчання.

4 Чому самоувага

У цьому розділі ми порівнюємо різні аспекти шарів самоуваги з рекурентними та згортковими шарами, які зазвичай використовують для відображення однієї послідовності представлень символів змінної довжини $(x_1, ..., x_n)$ в іншу послідовність тієї самої довжини $(z_1, ..., z_n)$, де $x_i, z_i \in \mathbb{R}^d$, наприклад у прихованому шарі типового кодувальника чи декодувальника перетворення послідовностей. Обґрунтовуючи використання самоуваги, ми розглядаємо три бажані властивості.

Перша — загальна обчислювальна складність одного шару. Друга — обсяг обчислень, який можна розпаралелити, вимірюваний мінімальною необхідною кількістю послідовних операцій.

Третя — довжина шляху між далекими залежностями в мережі. Навчання далеких залежностей є ключовою проблемою багатьох задач перетворення послідовностей. Один із головних чинників, що впливають на здатність навчатися таких залежностей, — довжина шляхів, які мають проходити прямі та зворотні сигнали в мережі. Що коротші ці шляхи між довільними позиціями вхідної та вихідної послідовностей, то легше навчатися далеких залежностей [12]. Тому ми також порівнюємо максимальну довжину шляху між довільними двома вхідними та вихідними позиціями в мережах із різними типами шарів.

Як зазначено в табл. 1, шар самоуваги з'єднує всі позиції сталою кількістю послідовно виконуваних операцій, тоді як рекурентний шар потребує $O(n)$ послідовних операцій. З погляду обчислювальної складності шари самоуваги швидші за рекурентні, коли довжина послідовності n менша за розмірність представлення d. Це найчастіше справджується для представлень речень, які використовують найкращі моделі машинного перекладу, зокрема представлень фрагментами слів [38] і парами байтів [31]. Для підвищення обчислювальної продуктивності в задачах із дуже довгими послідовностями самоувагу можна обмежити лише околом розміру r у вхідній послідовності з центром у відповідній вихідній позиції. Це збільшить максимальну довжину шляху до $O(n/r)$. Ми плануємо докладніше дослідити цей підхід у майбутніх роботах.

Таблиця 1: Максимальні довжини шляхів, складність одного шару та мінімальна кількість послідовних операцій для різних типів шарів. n — довжина послідовності, d — розмірність представлення, k — розмір ядра згорток, а r — розмір околу в обмеженій самоувазі.

Тип шару	Складність шару	Послідовні операції	Максимальна довжина шляху
Самоувага	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Рекурентний	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Згортковий	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Самоувага (обмежена)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Один згортковий шар із шириною ядра $k < n$ не з'єднує всі пари вхідних і вихідних позицій. Для цього потрібен стек із $O(n/k)$ згорткових шарів у разі суцільних ядер або $O(log_k(n))$ у разі розширених згорток [18], що збільшує довжину найдовших шляхів між будь-якими двома позиціями мережі. Згорткові шари загалом у k разів обчислювально дорожчі за рекурентні. Проте роздільні згортки [6] істотно зменшують складність — до $O(k \cdot n \cdot d + n \cdot d^2)$. Навіть за $k = n$ складність роздільної згортки дорівнює складності поєднання шару самоуваги з поточковим шаром прямого поширення — саме цей підхід ми застосовуємо у своїй моделі.

Додатковою перевагою самоуваги може бути краща інтерпретованість моделей. Ми досліджуємо розподіли уваги наших моделей і наводимо та обговорюємо приклади в додатку. Окремі голови уваги не лише явно навчаються виконувати різні задачі: багато з них, схоже, демонструють поведінку, пов'язану із синтаксичною та семантичною структурою речень.

5 Навчання

У цьому розділі описано режим навчання наших моделей.

5.1 Навчальні дані та формування пакетів

Ми навчалися на стандартному англо-німецькому наборі даних WMT 2014, що містить близько 4.5 мільйона пар речень. Речення кодували за допомогою кодування парами байтів [3] зі спільним для вихідної та цільової мов словником приблизно на 37000 токенів. Для англо-французького перекладу ми використали значно більший англо-французький набір WMT 2014, що містить 36M речень, і поділили токени на фрагменти слів зі словником на 32000 одиниць [38]. Пари речень об'єднували в пакети за приблизною довжиною послідовностей. Кожен навчальний пакет містив набір пар речень із приблизно 25000 токенів вихідної та 25000 токенів цільової мови.

5.2 Обладнання та графік навчання

Ми навчали моделі на одній машині з 8 графічними процесорами NVIDIA P100. Для базових моделей із гіперпараметрами, описаними в цій статті, кожен крок навчання тривав близько 0.4 секунди. Базові моделі навчали загалом 100,000 кроків, або 12 годин. Для великих моделей (описаних в останньому рядку табл. 3) час кроку становив 1.0 секунди. Великі моделі навчали 300,000 кроків (3.5 дня).

5.3 Оптимізатор

Ми використовували оптимізатор Adam [20] з $\beta_1 = 0.9$, $\beta_2 = 0.98$ та $\epsilon = 10^{-9}$. У процесі навчання швидкість навчання змінювали за формулою:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \tag{3}$$

Це відповідає лінійному збільшенню швидкості навчання протягом перших $warmup_steps$ кроків і подальшому її зменшенню обернено пропорційно квадратному кореню з номера кроку. Ми використовували $warmup_steps = 4000$.

Таблиця 2: Transformer досягає вищих показників BLEU, ніж попередні найкращі моделі, на тестах newstest2014 для англо-німецького й англо-французького перекладу за значно менших витрат на навчання.

Модель	BLEU		Витрати навчання (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk ансамбль [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL ансамбль [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S ансамбль [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (базова модель)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (велика)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

5.4 Регуляризація

Під час навчання ми застосовуємо три типи регуляризації:

Відсів у залишкових з'єднаннях Ми застосовуємо відсів (dropout) [33] до виходу кожного підшару перед його додаванням до входу підшару та нормалізацією. Крім того, застосовуємо відсів до сум вкладень і позиційних кодувань у стеках кодувальника та декодувальника. Для базової моделі використовуємо ймовірність $P_{drop} = 0.1$.

Згладжування міток Під час навчання ми використовували згладжування міток зі значенням $\epsilon_{ls} = 0.1$ [36]. Це погіршує перплексію, оскільки модель навчається бути менш упевненою, але підвищує точність і показник BLEU.

6 Результати

6.1 Машинний переклад

У задачі англо-німецького перекладу WMT 2014 велика модель Transformer (Transformer (велика) у табл. 2) перевершує найкращі раніше опубліковані моделі, включно з ансамблями, більш ніж на 2.0 BLEU, встановлюючи новий найкращий показник BLEU 28.4. Конфігурацію цієї моделі наведено в останньому рядку табл. 3. Навчання тривало 3.5 дня на 8 графічних процесорах P100. Навіть наша базова модель перевершує всі раніше опубліковані моделі й ансамблі за значно менших витрат на навчання, ніж у будь-якої конкурентної моделі.

У задачі англо-французького перекладу WMT 2014 наша велика модель досягає BLEU 41.0, перевершуючи всі раніше опубліковані одиночні моделі за витрат на навчання менш як $1/4$ від витрат попередньої найкращої моделі. Модель Transformer (велика), навчена для англо-французького перекладу, використовувала ймовірність відсіву $P_{drop} = 0.1$ замість 0.3.

Для базових моделей ми використовували одну модель, отриману усередненням останніх 5 контрольних точок, збережених з інтервалом у 10 хвилин. Для великих моделей усереднювали останні 20 контрольних точок. Ми застосовували променевий пошук із шириною променя 4 та штрафом за довжину $\alpha = 0.6$ [38]. Ці гіперпараметри обрали після експериментів на валідаційному наборі. Максимальну довжину виходу під час виведення встановили як довжину входу + 50, але за можливості завершували раніше [38].

Табл. 2 підсумовує наші результати й порівнює якість перекладу та витрати на навчання з іншими архітектурами моделей з літератури. Кількість операцій із рухомою комою, використаних для навчання моделі, ми оцінюємо як добуток часу навчання, кількості використаних графічних проце-

Таблиця 3: Варіанти архітектури Transformer. Ненаведені значення збігаються з базовою моделлю. Усі метрики отримано на валідаційному наборі англо-німецького перекладу newstest2013. Наведені перплексії розраховано на фрагмент слова за нашим кодуванням парами байтів; їх не слід порівнювати з перплексіями на слово.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	кроки навч.	PPL (вал.)	BLEU (вал.)	парам. $\times 10^6$
базова	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	позиційні вкладення замість синусоїд									4.92	25.7	
велика	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

сорів та оцінки сталої продуктивності кожного процесора в операціях із рухомою комою одинарної точності[5].

6.2 Варіанти моделі

Щоб оцінити важливість різних складових Transformer, ми різними способами змінювали базову модель, вимірюючи зміну якості англо-німецького перекладу на валідаційному наборі newstest2013. Ми використовували променевий пошук, описаний у попередньому розділі, але без усереднення контрольних точок. Ці результати наведено в табл. 3.

У рядках (A) табл. 3 ми змінюємо кількість голів уваги та розмірності ключів і значень уваги, зберігаючи сталий обсяг обчислень, як описано в розділі 3.2.2. Хоча одноголова увага поступається найкращому налаштуванню на 0.9 BLEU, надмірна кількість голів також знижує якість.

У рядках (B) табл. 3 бачимо, що зменшення розміру ключа уваги d_k погіршує якість моделі. Це вказує, що визначати сумісність непросто і що складніша за скалярний добуток функція сумісності може бути корисною. Крім того, у рядках (C) та (D) бачимо, що, як і очікувалося, більші моделі кращі, а відсів дуже допомагає уникати перенавчання. У рядку (E) замінюємо синусоїдне позиційне кодування навченими позиційними вкладеннями [9] й отримуємо майже ідентичні до базової моделі результати.

6.3 Синтаксичний аналіз англійської за складниками

Щоб перевірити, чи здатен Transformer узагальнювати на інші задачі, ми провели експерименти із синтаксичного аналізу англійської за складниками. Ця задача має специфічні труднощі: вихід підпорядковується сильним структурним обмеженням і є значно довшим за вхід. Крім того, рекурентні моделі перетворення послідовностей не змогли досягти найкращих результатів у режимах із малою кількістю даних [37].

[5]Ми використовували значення 2.8, 3.7, 6.0 та 9.5 TFLOPS для K80, K40, M40 та P100 відповідно.

Таблиця 4: Transformer добре узагальнює на синтаксичний аналіз англійської за складниками (результати на розділі 23 WSJ)

Аналізатор	Навчання	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	лише WSJ, дискримінативне	88.3
Petrov et al. (2006) [29]	лише WSJ, дискримінативне	90.4
Zhu et al. (2013) [40]	лише WSJ, дискримінативне	90.4
Dyer et al. (2016) [8]	лише WSJ, дискримінативне	91.7
Transformer (4 шари)	лише WSJ, дискримінативне	91.3
Zhu et al. (2013) [40]	з частковим наглядом	91.3
Huang & Harper (2009) [14]	з частковим наглядом	91.3
McClosky et al. (2006) [26]	з частковим наглядом	92.1
Vinyals & Kaiser el al. (2014) [37]	з частковим наглядом	92.1
Transformer (4 шари)	з частковим наглядом	92.7
Luong et al. (2015) [23]	багатозадачне	93.0
Dyer et al. (2016) [8]	генеративне	93.3

Ми навчали 4-шаровий Transformer з $d_{model} = 1024$ на частині Wall Street Journal (WSJ) корпусу Penn Treebank [25], що містить близько 40K навчальних речень. Також навчали його в режимі з частковим наглядом, використовуючи більші корпуси high-confidence та BerkleyParser приблизно із 17M речень [37]. У режимі лише WSJ використовували словник на 16K токенів, а в режимі з частковим наглядом — на 32K токенів.

Ми провели лише невелику кількість експериментів для вибору ймовірностей відсіву, як в увазі, так і в залишкових з'єднаннях (розділ 5.4), швидкостей навчання та ширини променя на валідаційному наборі розділу 22. Усі інші параметри залишилися такими, як у базовій моделі англо-німецького перекладу. Під час виведення максимальну довжину виходу збільшили до довжини входу + 300. Для обох режимів, лише WSJ та з частковим наглядом, використовували ширину променя 21 і $\alpha = 0.3$.

Наші результати в табл. 4 показують, що, попри відсутність спеціального налаштування для цієї задачі, модель працює напрочуд добре, даючи кращі результати, ніж усі раніше опубліковані моделі, за винятком Recurrent Neural Network Grammar [8].

На відміну від рекурентних моделей перетворення послідовностей [37], Transformer перевершує BerkeleyParser [29] навіть за навчання лише на навчальному наборі WSJ із 40K речень.

7 Висновки

У цій роботі ми представили Transformer — першу модель перетворення послідовностей, повністю побудовану на увазі, яка замінює рекурентні шари, найчастіше використовувані в архітектурах кодувальника-декодувальника, багатоголовою самоувагою.

Для задач перекладу Transformer можна навчати значно швидше, ніж архітектури на основі рекурентних або згорткових шарів. В обох задачах WMT 2014, англо-німецького й англо-французького перекладу, ми досягаємо нового найкращого результату. У першій задачі наша найкраща модель перевершує навіть усі раніше опубліковані ансамблі.

Ми з ентузіазмом дивимося на майбутнє моделей на основі уваги та плануємо застосувати їх до інших задач. Ми плануємо розширити Transformer на задачі з іншими, крім тексту, вхідними й вихідними модальностями та дослідити локальні, обмежені механізми уваги для ефективного опрацювання великих входів і виходів, зокрема зображень, аудіо та відео. Ще одна наша дослідницька мета — зменшити обсяг послідовних обчислень під час генерування.

Код, який ми використовували для навчання та оцінювання моделей, доступний за адресою `https://github.com/tensorflow/tensor2tensor`.

Подяки Ми вдячні Nal Kalchbrenner та Stephan Gouws за плідні зауваження, виправлення та натхнення.

Література

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Нормалізація шару. *Препринт arXiv arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Нейронний машинний переклад шляхом спільного навчання вирівнювання та перекладу. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Масштабне дослідження архітектур нейронного машинного перекладу. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Мережі довгої короткочасної пам'яті для машинного читання. *Препринт arXiv arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Навчання представлень фраз за допомогою рекурентного кодувальника-декодувальника для статистичного машинного перекладу. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: глибоке навчання з роздільними за глибиною згортками. *Препринт arXiv arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Емпіричне оцінювання вентильних рекурентних нейронних мереж у моделюванні послідовностей. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Граматики рекурентних нейронних мереж. У *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Згорткове навчання перетворення послідовностей. *Препринт arXiv arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Генерування послідовностей рекурентними нейронними мережами. *Препринт arXiv arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Глибоке залишкове навчання для розпізнавання зображень. У *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, с. 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Потік градієнта в рекурентних мережах: складність навчання довготривалих залежностей, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Довга короткочасна пам'ять. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Самонавчання граматик PCFG із прихованими анотаціями для різних мов. У *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, с. 832–841. ACL, серпень 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Дослідження меж мовного моделювання. *Препринт arXiv arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Чи може активна пам'ять замінити увагу? У *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Нейронні GPU навчаються алгоритмів. У *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Нейронний машинний переклад за лінійний час. *Препринт arXiv arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Мережі структурованої уваги. У *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: метод стохастичної оптимізації. У *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Прийоми факторизації для мереж LSTM. *Препринт arXiv arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. Структуроване самоуважне векторне вкладення речення. *Препринт arXiv arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Багатозадачне навчання перетворення послідовностей. *Препринт arXiv arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Ефективні підходи до нейронного машинного перекладу на основі уваги. *Препринт arXiv arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Побудова великого анотованого корпусу англійської мови: Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Ефективне самонавчання для синтаксичного аналізу. У *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, с. 152–159. ACL, червень 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. Розкладна модель уваги. У *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. Глибока модель з навчанням з підкріпленням для абстрактивного підсумовування. *Препринт arXiv arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Навчання точного, компактного та інтерпретованого анотування дерев. У *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, с. 433–440. ACL, липень 2006.

[30] Ofir Press and Lior Wolf. Використання вихідного векторного вкладення для поліпшення мовних моделей. *Препринт arXiv arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Нейронний машинний переклад рідкісних слів із підслівними одиницями. *Препринт arXiv arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Надзвичайно великі нейронні мережі: шар суміші експертів із розрідженим вентилюванням. *Препринт arXiv arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Відсів: простий спосіб запобігати перенавчанню нейронних мереж. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Наскрізні мережі пам'яті. У C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, ред., *Advances in Neural Information Processing Systems 28*, с. 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Навчання перетворення послідовностей нейронними мережами. У *Advances in Neural Information Processing Systems*, с. 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Переосмислення архітектури Inception для комп'ютерного зору. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Граматика як іноземна мова. У *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Система нейронного машинного перекладу Google: подолання розриву між людським і машинним перекладом. *Препринт arXiv arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Глибокі рекурентні моделі зі з'єднаннями прямого передавання для нейронного машинного перекладу. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Швидкий і точний синтаксичний аналіз за складниками методом зсуву-редукції. У *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, с. 434–443. ACL, серпень 2013.

Візуалізації уваги

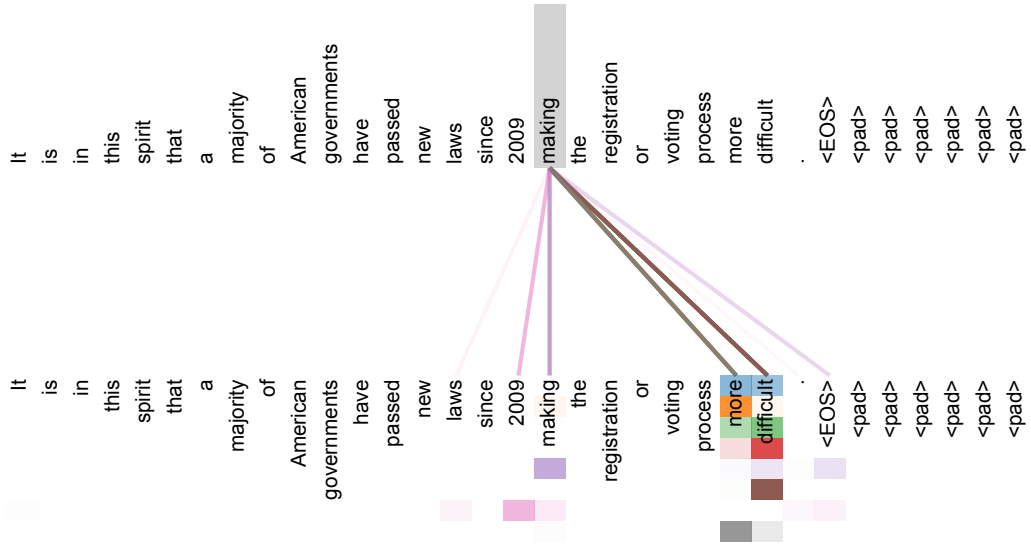


Рисунок 3: Приклад того, як механізм уваги відстежує далекі залежності в самоувазі кодувальника на шарі 5 із 6. Багато голів уваги звертають увагу на віддалену залежність дієслова ‘making’, завершуючи фразу ‘making...more difficult’. Тут показано увагу лише для слова ‘making’. Різні кольори позначають різні голови. Найкраще переглядати в кольорі.

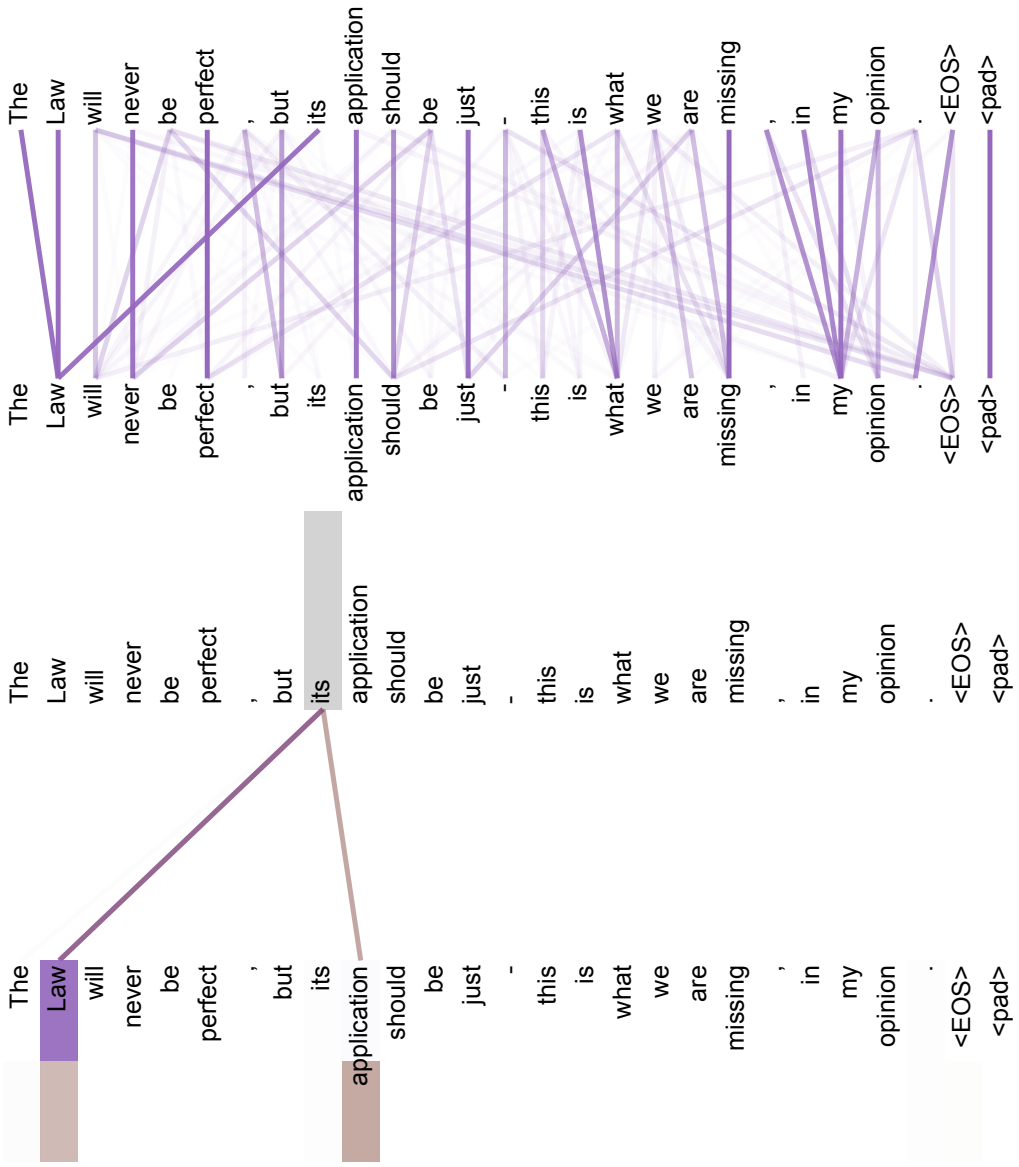


Рисунок 4: Дві голови уваги, також на шарі 5 із 6, які, очевидно, беруть участь у розв’язанні анафори. Угорі: повна увага голови 5. Унизу: ізольована увага лише від слова ‘its’ для голів 5 і 6. Зверніть увагу, що для цього слова розподіли уваги дуже зосереджені.

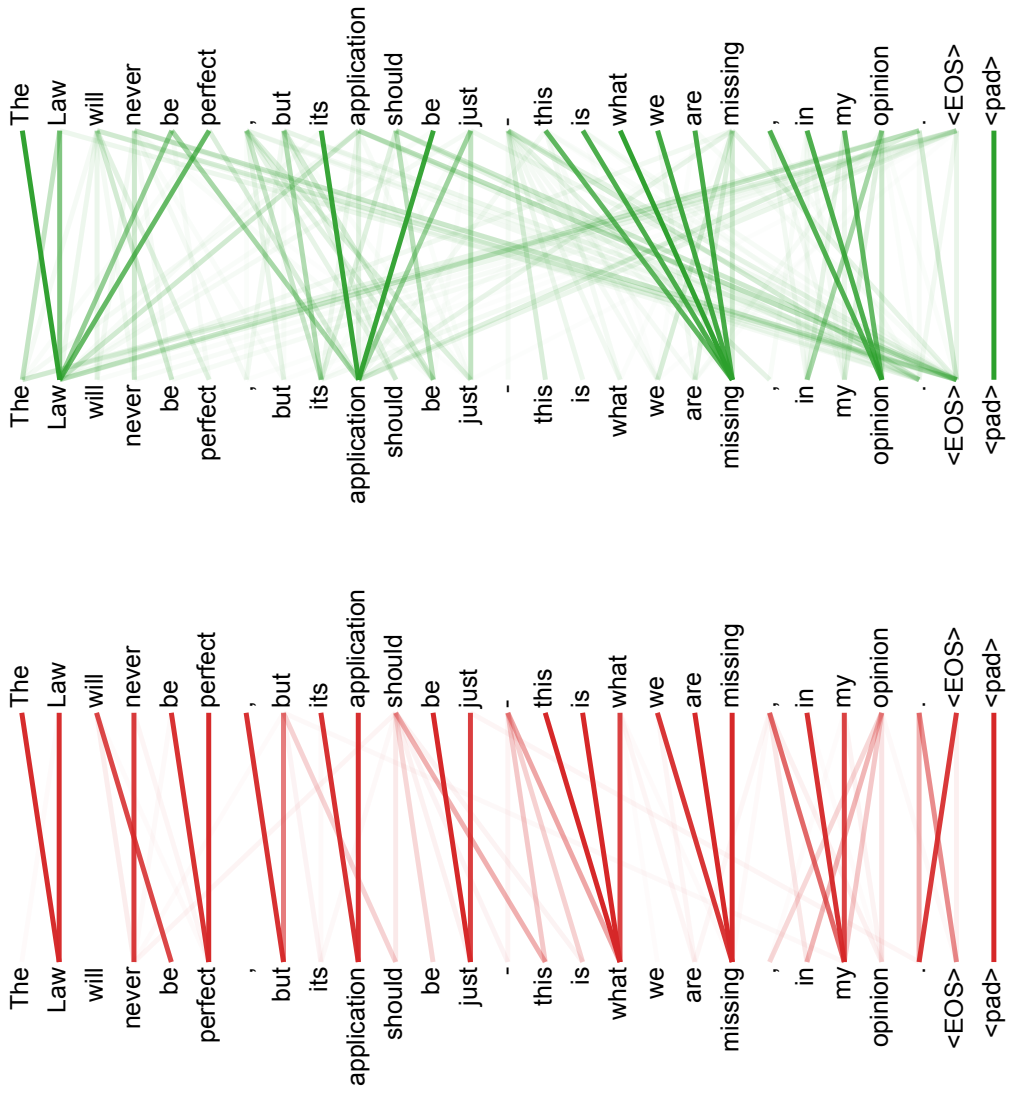


Рисунок 5: Поведінка багатьох голів уваги, схоже, пов’язана зі структурою речення. Вище наведено два такі приклади для двох різних голів самоуваги кодувальника на шарі 5 із 6. Голови явно навчилися виконувати різні задачі.