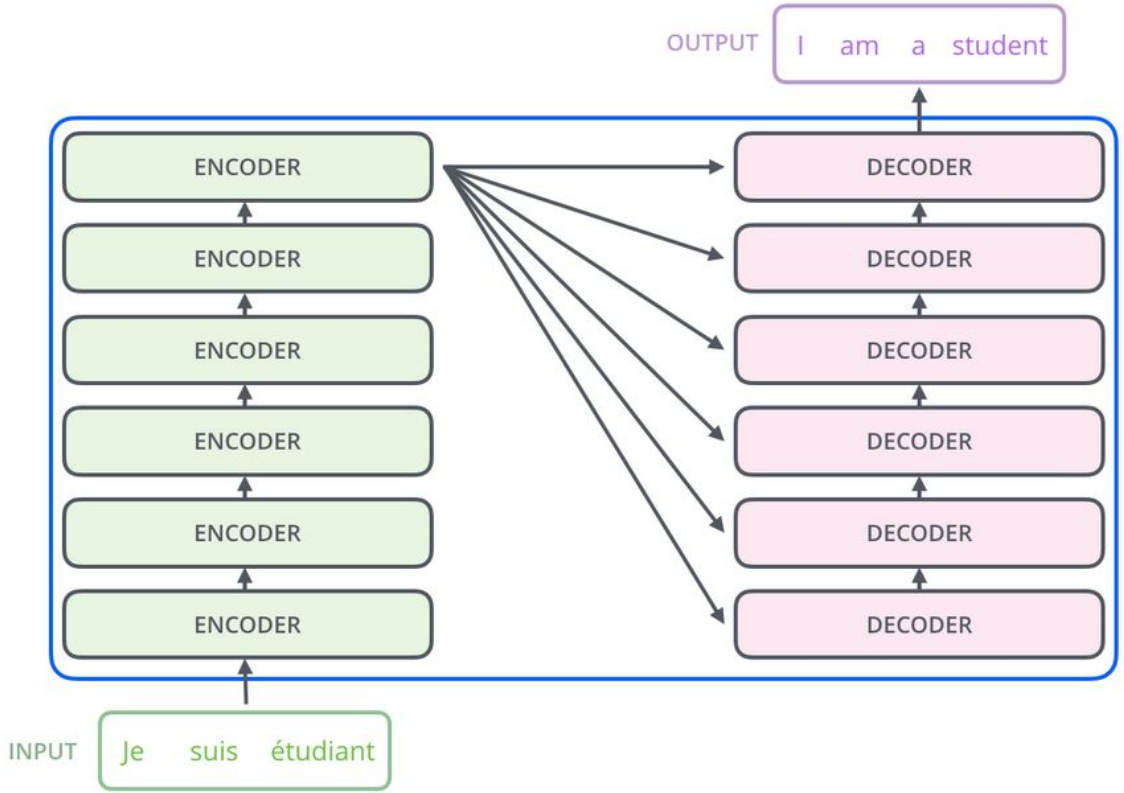
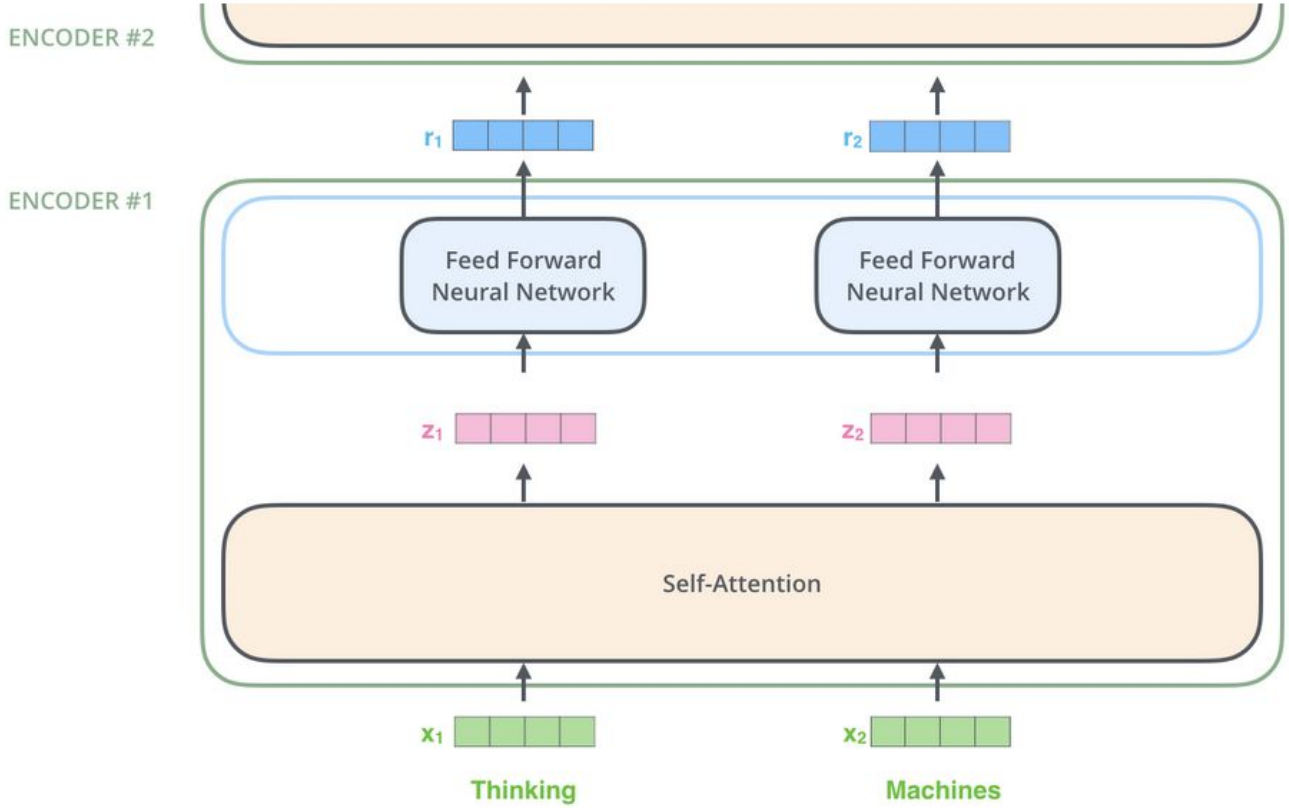
Transformers are RNNs

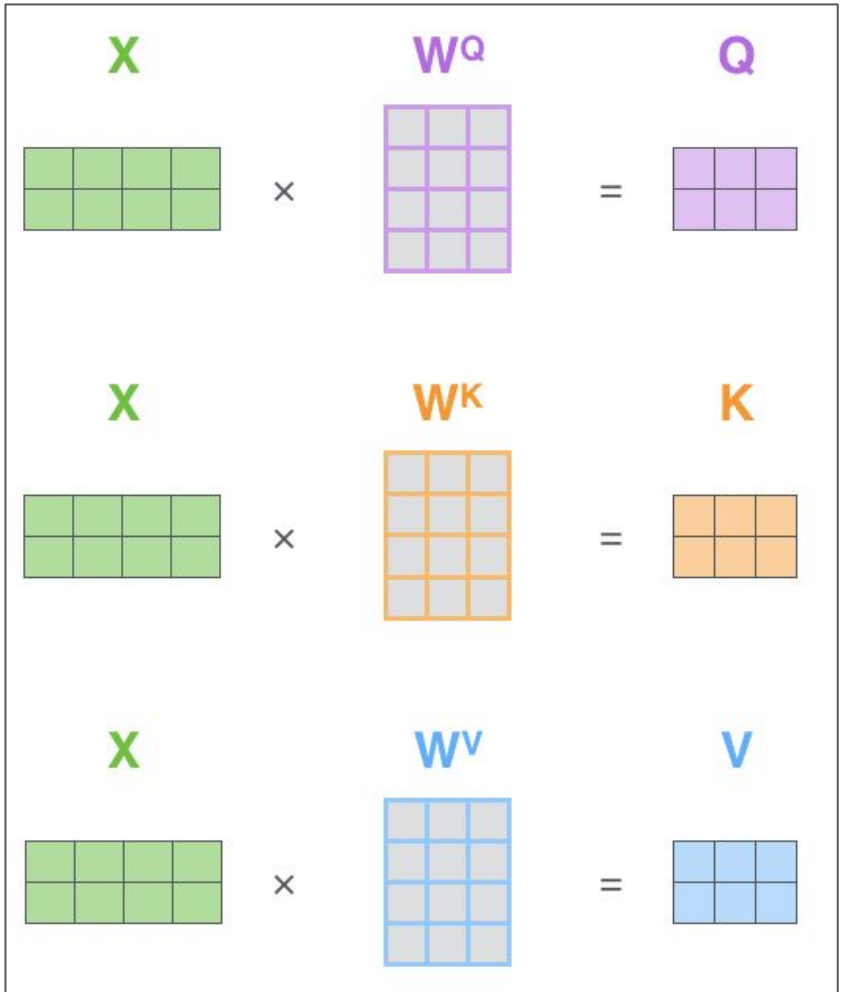
Пахалко Илья, 181

Vanilla Transformer



Vanilla Transformer





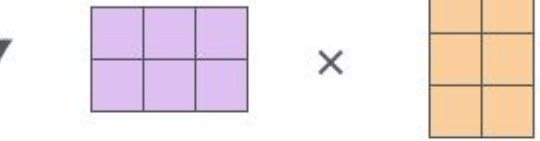
	Размеры:	Сложность:
Q	$(N,F) \times (F,D) = (N,D)$	$O(NFD)$
K	$(N,F) \times (F,D) = (N,D)$	$O(NFD)$
V	$(N,F) \times (F,M) = (N,M)$	$O(NFM)$

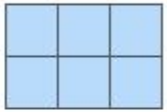
(N,D) (D,N) (N,M)

Q K^T V

(построчный)

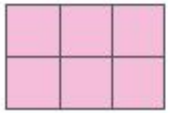
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V$$





V’

=



$$O(N^2 D) + O(N^2 M) = O(N^2 \max(D, M))$$

Построчная запись

$$V_i' = \frac{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \mathrm{sim}\left(Q_i, K_j\right)}.$$

$$\mathrm{sim}\left(q, k\right) = \exp\left(\frac{q^T k}{\sqrt{D}}\right)$$

V_j, V_i', Q_i, K_j - соответствующие строки

оказывается, можем взять

$$sim(x, y) = K(x, y) = \langle \phi(x), \phi(y) \rangle = \phi(x)^T \phi(y),$$

$$K(x, y) : \mathbb{R}^{2 \times F} \rightarrow \mathbb{R}_+$$

(Применение спрямляющего отображения - построчное)

$$V_i' = \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)} = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}$$

Запутались в нотации?..

(Применение спрямляющего отображения - построчное)

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)} = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$$

числитель:

$$\left(\phi(Q)\phi(K)^T\right)V = \phi(Q)\left(\phi(K)^T V\right)$$

“знаменатель” - диагональная матрица, на обратную к которой домножаем числитель:

$$diag((\phi(Q)\phi(K)^T)\vec{\mathbb{1}}) = diag(\phi(Q)(\phi(K)^T\vec{\mathbb{1}}))$$

где $\vec{\mathbb{1}}$ - вектор единиц размера N

Запутались в нотации?..

Что нам это даёт?

$$\left(\phi\left(Q\right)\phi\left(K\right)^{T}\right)V = \phi\left(Q\right)\left(\phi\left(K\right)^{T}V\right)$$

(N,C) (C,N) (N,M) (C,M)

Сложность: $O(NCM) + O(NCM) = O(NCM)$

где C - размерность спрямляющего пространства

Что это означает?

- Не имеет смысла использовать экспоненциальное (гауссово) ядро - его спрямляющее пространство - бесконечномерное:

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) =$$

$$= \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|z\|^2}{2\sigma^2}\right)\sum_{k=0}^{\infty}\frac{\langle x,z\rangle^k}{k!\sigma^{2k}}$$

Что это означает?

- Квадратичное ядро имеет спрямляющее пространство размерности D^2

$$K(x, y) = \left(\sum_{i=1}^{n} x_i y_i + c \right)^2$$

$$\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2}x_n x_{n-1}, \ldots, \sqrt{2}x_n x_1, \sqrt{2}x_{n-1} x_{n-2}, \ldots, \sqrt{2}x_{n-1} x_1, \ldots, \sqrt{2}x_2 x_1, \sqrt{2c}x_n, \ldots, \sqrt{2c}x_1, c \rangle$$

Что это означает?

- Квадратичное ядро имеет спрямляющее пространство размерности D^2

$$K(x, y) = \left(\sum_{i=1}^{n} x_i y_i + c \right)^2$$

$$\mathcal{O}\left(N D^2 M\right)$$

хорошо, когда $N > D^2$

$$\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2} x_n x_{n-1}, \ldots, \sqrt{2} x_n x_1, \sqrt{2} x_{n-1} x_{n-2}, \ldots, \sqrt{2} x_{n-1} x_1, \ldots, \sqrt{2} x_2 x_1, \sqrt{2c} x_n, \ldots, \sqrt{2c} x_1, c \rangle$$

Что было использовано:

$$\phi(x) = ELU(x) + 1$$

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$

$C = D$, сложность: $O(NDM)$

Causal Masking

$$V_i' = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right)}$$

Causal Masking

$$V_i' = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right)}$$

$$S_i = \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T,\ Z_i = \sum_{j=1}^{i} \phi\left(K_j\right) \Rightarrow V_i' = \frac{\phi\left(Q_i\right)^T S_i}{\phi\left(Q_i\right)^T Z_i}$$

Causal Masking

$$V_i' = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right)}$$

$$S_i = \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T, \; Z_i = \sum_{j=1}^{i} \phi\left(K_j\right) \Rightarrow V_i' = \frac{\phi\left(Q_i\right)^T S_i}{\phi\left(Q_i\right)^T Z_i}$$

(C,M)

=(D,M)

Наивное хранение частичных сумм увеличивает затраты по памяти в $\max\left(D, M\right)$ раз!

Рекуррентный подсчёт градиентов

$$\nabla_{\phi(Q_i)}\mathcal{L} = \nabla_{\bar{V}_i}\mathcal{L}\left(\sum_{j=1}^{i}\phi\left(K_j\right)V_j^T\right)^T,$$

$$\nabla_{\phi(K_i)}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)V_i\ ,$$

$$\nabla_{V_i}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)^T\phi\left(K_i\right)$$

Рекуррентный подсчёт градиентов

function backward$(\phi(Q), \phi(K), V, G)$**:**
  /* G is the gradient of the loss with respect to the output of forward */
  $S \leftarrow 0, \nabla_{\phi(Q)}\mathcal{L} \leftarrow 0$
  for $i = 1, \ldots, N$ **do**
    $S \leftarrow S + \phi(K_i)\, V_i^T$
    $\nabla_{\phi(Q_i)}\mathcal{L} \leftarrow G_i S^T$ equation 13
  end
  $S \leftarrow 0, \nabla_{\phi(K)}\mathcal{L} \leftarrow 0, \nabla_V \mathcal{L} \leftarrow 0$
  for $i = N, \ldots, 1$ **do**
    $S \leftarrow S + \phi(Q_i)\, G_i^T$
    $\nabla_{V_i}\mathcal{L} \leftarrow S^T \phi(K_i)$ equation 15
    $\nabla_{\phi(K_i)}\mathcal{L} \leftarrow S V_i$ equation 14
  end
  return $\nabla_{\phi(Q)}\mathcal{L}, \nabla_{\phi(K)}\mathcal{L}, \nabla_V \mathcal{L}$
end

Transformers are… RNNs

$$
\begin{aligned}
s_0 &= 0, \\
z_0 &= 0, \\
s_i &= s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \\
z_i &= z_{i-1} + \phi\left(x_i W_K\right), \\
y_i &= f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right).
\end{aligned}
$$

Quick Recap

- На стадии обучения линейный трансформер остаётся высокопараллелизуемым (?)
- На стадии тестирования бутылочное горлышко остаётся, но за счёт линейной асимптотики модель работает значительно быстрее

Насколько быстрее?

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
Stateful-softmax	0.621	7.56	(16.8×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

(a) Image generation on MNIST

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
Stateful-softmax	3.47	0.32	(80×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

(b) Image generation on CIFAR-10

Table 4: Comparison of autoregressive image generation throughput of MNIST and CIFAR-10 images. The experiment can be found in § 4.2 in the main paper. For stateful-softmax we save the keys and values and reuse them for predicting the next element. A detailed description of this extra baseline can be found in § C.1.

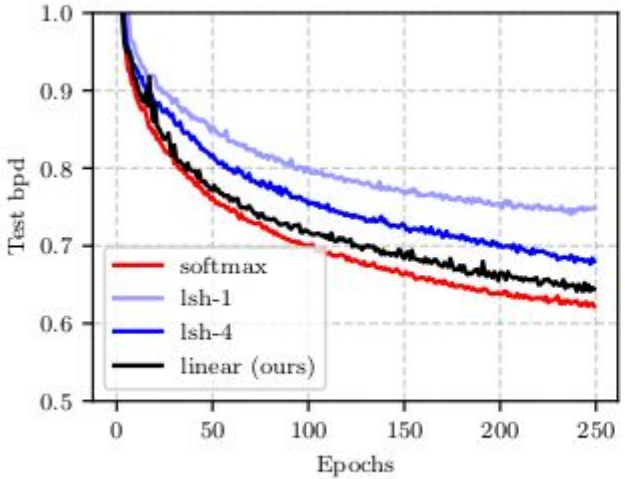
Method	Seconds (CPU)		Seconds (GPU)	
Softmax	72.6	(13.2×)	10.2	(1.4×)
Stateful-softmax	7.4	(1.3×)	10.4	(1.42×)
LSH-1	46.0	(8.3×)	19.2	(2.6×)
LSH-4	112.0	(20×)	55.8	(7.6×)
Linear (ours)	**5.5**	(1×)	**7.3**	(1×)

(a) Image generation on MNIST

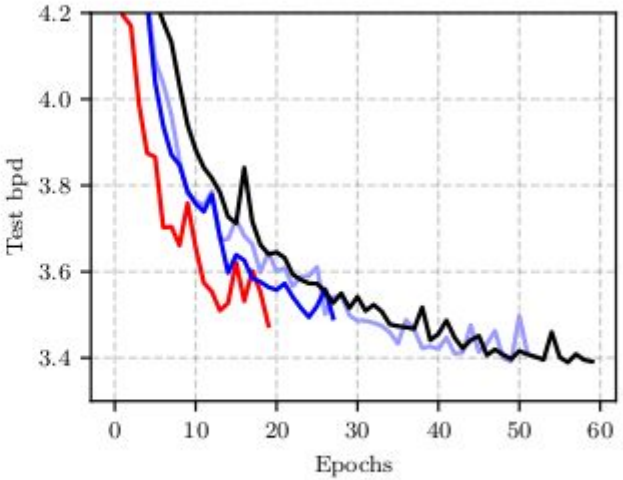
Method	Seconds (CPU)		Seconds (GPU)	
Softmax	8651.4	(191.8×)	300.1	(4.9×)
Stateful-softmax	71.9	(1.6×)	70.4	(1.14×)
LSH-1	2318.9	(51.4×)	221.6	(3.6×)
LSH-4	5263.7	(116.7×)	683.9	(11.1×)
Linear (ours)	**45.1**	(1×)	**61.3**	(1×)

(b) Image generation on CIFAR-10

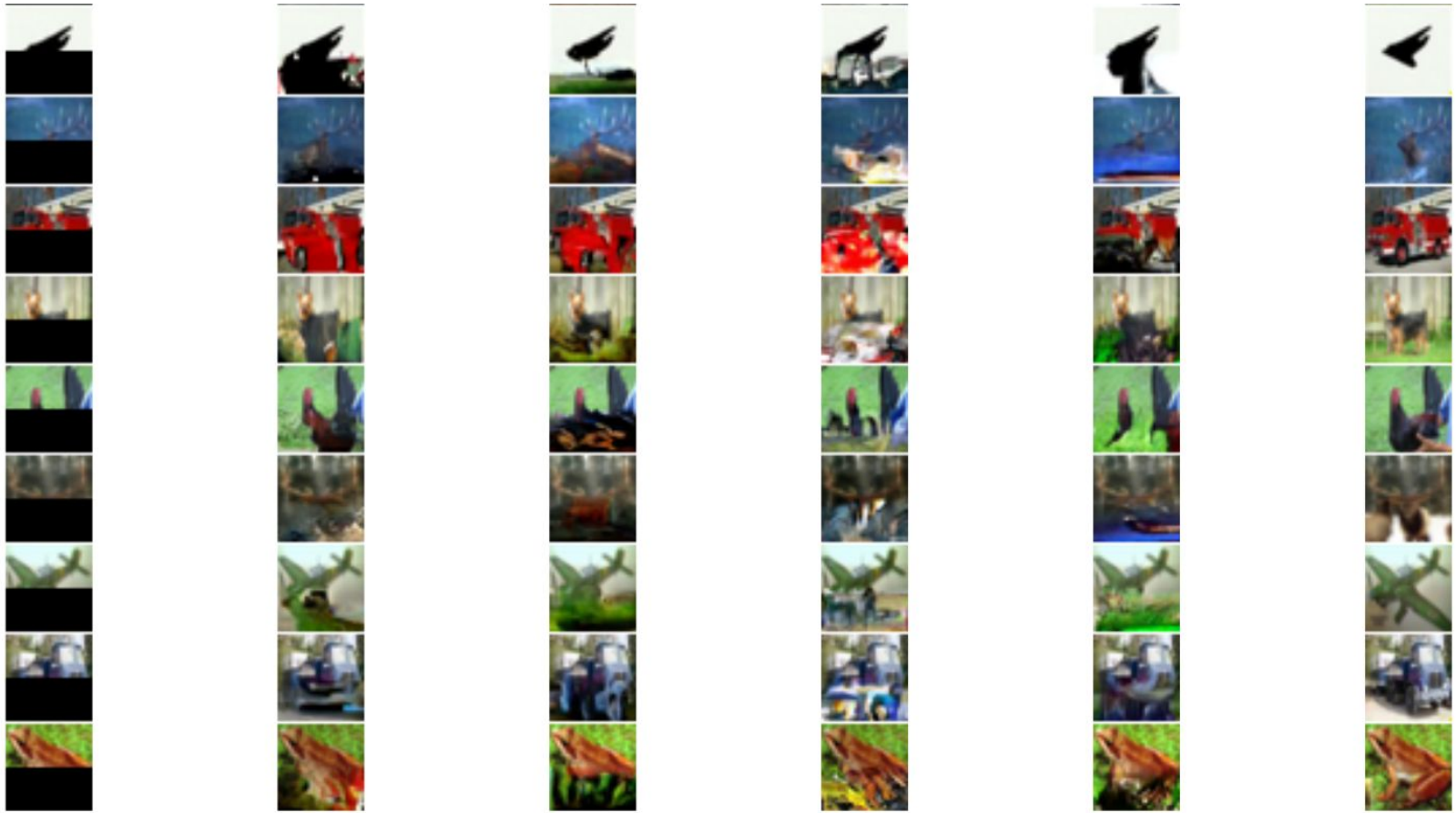
Table 5: Comparison of the time required to generate a single image with autoregressive transformers on MNIST and CIFAR-10. We run all methods with a batch size of 1 both on CPU and GPU and report the total time in seconds. For all numbers in the table, lower is better.



(a) MNIST



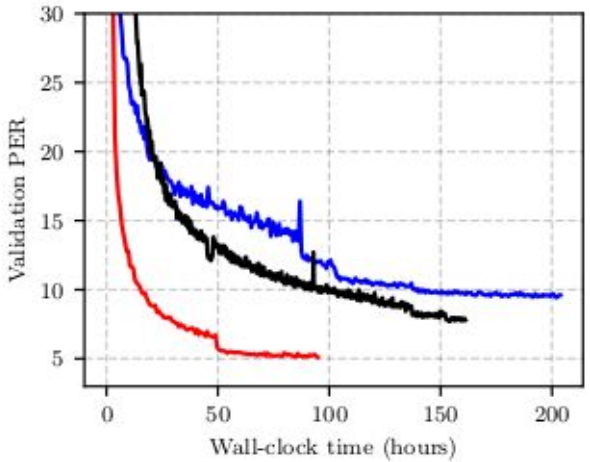
(b) CIFAR-10



(a) Occluded (b) Softmax (c) Linear (ours) (d) LSH-1 (e) LSH-4 (f) Original

Figure 9: CIFAR-10 image completions from all trained transformer models. See § 4.2.2 in the main paper.

Method	Validation PER	Time/epoch (s)
Bi-LSTM	10.94	1047
Softmax	5.12	2711
LSH-4	9.33	2250
Linear (ours)	8.08	**824**

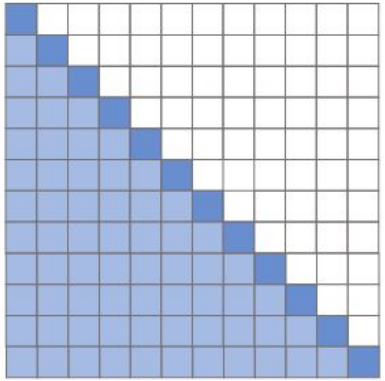


(c) Speech Recognition

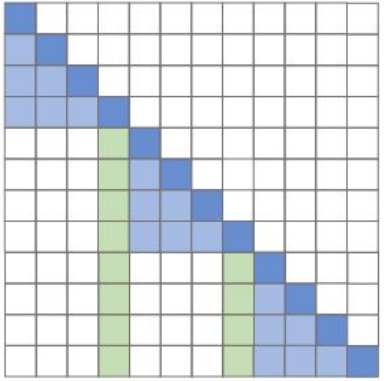
Как ещё ускорять?

- Сокращать область внимания до ширины окна = b. Внимание считается по N/b блокам, итоговая асимптотика O(b^2 * n/b) = O(nb)
- Считать внимание разреженно, в зависимости от индексов (Sparse Transformer)
- Считать внимание разреженно, используя локально-чувствительное хеширование (Reformer)

Sparse Transformer, память O(NlogN)



(a) Transformer



(b) Sparse Transformer

Figure 4: Illustration of patterns of the attention matrix for dense self-attention in Transformers and sparse fixed attention in Sparse Transformers.

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } \lfloor j/N \rfloor = \lfloor i/N \rfloor \\ 0 & \text{otherwise} \end{cases}$$

Или

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } (i-j) \mod N = 0 \\ 0 & \text{otherwise} \end{cases}$$

Использованные материалы:

- Оригинальная статья: https://arxiv.org/abs/2006.16236
- Обзор эффективных трансформеров: https://arxiv.org/abs/2009.06732
- Иллюстрированный трансформер: https://jalammar.github.io/illustrated-transformer/

Пояснение к нотации

Авторы в оригинальной статье применяют следующую нотацию:

за K_j обозначается j-я строка матрицы K, расположенная как **столбец.**

Соответственно, $\phi(K_j)^T$ - это образ j-ой строки, расположенный как **строка.**

Назад!