The Transformer

1. Self-attention

input sequence: $x_1, x_2 \dots x_T \quad x_t \in \mathbb{R}^d$

Queries: $q_t = Q x_t \quad Q \in \mathbb{R}^{d_q \times d}$

Keys: $k_t = K x_t \quad K \in \mathbb{R}^{d_k \times d}$ $\quad (d_q = d_k)$

Values: $v_t = V x_t \quad V \in \mathbb{R}^{d_v \times d}$

$\rightarrow$ trainable params

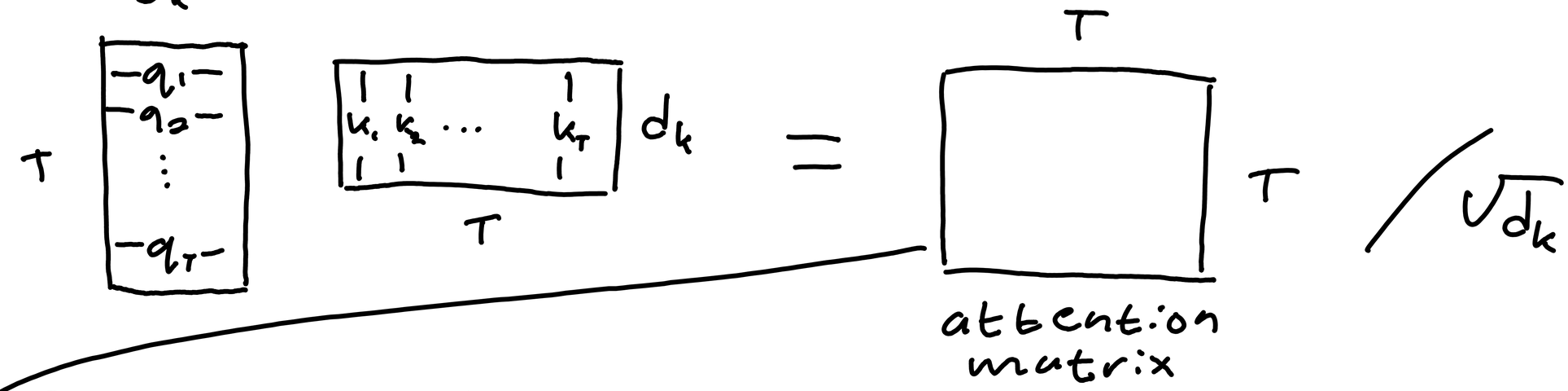
Comparison function:

$$a_{ij} = \alpha(q_i, k_j) = \frac{q_i^T k_j}{\sqrt{d_k}}$$

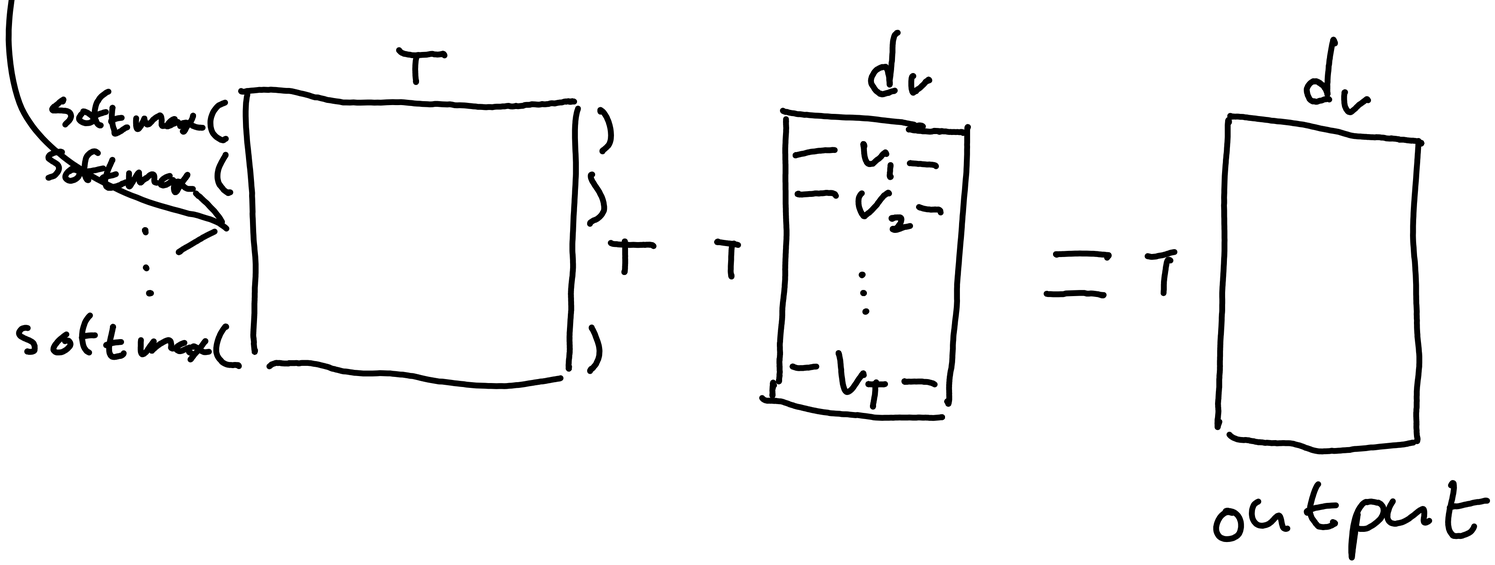
Reduction function:

softmax-weighted average of $v_1, \dots, v_T$

comparison function:



Reduction function



2. Position Embeddings

Self-attention is orderless (set operation)

a) Learned vector (embedding) for each sequence position (can't extrapolate)

b) Relative embeddings for each sequence offset up to some maximum offset

how? i) $\alpha(q_i, k_j) = \frac{q_i^T (k_j + \underbrace{p_{ij}}_{\text{learned vector} \in \mathbb{R}^{d_k}})}{\sqrt{d_k}}$

ii) $\alpha(q_i, k_j) = \frac{q_i^T k_j}{\sqrt{d_k}} + \underbrace{p_{ij}}_{\text{learned scalar}}$

iii) $\alpha(q_i, k_j) = \frac{q_i^T k_j}{\sqrt{d_k}} + \lambda(i - j)$

↳ learned scalar

3. Multi-headed self-attention

Rather than having a single self-attention, we have H in parallel, then we aggregate the result.

$$h^{(i)} = \text{self attention}(Q^{(i)}, K^{(i)}, V^{(i)})$$

↳ sequence

↳ different parameters for each head

$$\text{output} = \bigcirc \begin{bmatrix} h^{(1)} \\ h^{(2)} \\ \vdots \\ h^{(H)} \end{bmatrix}$$

↳ output matrix $\mathbb{R}^{d_v \times H d_v}$

4. Layer norm

Given a length-T sequence $h_1, h_2, \ldots, h_T \in \mathbb{R}^d$

$$\mu_t = \frac{1}{d} \sum_{i=1}^{d} h_{t,i}$$

$$\sigma_t = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (h_{t,i} - \mu_t)^2}$$

$$\text{LayerNorm}(h)_t = \frac{\gamma}{\sigma_t}(h_t - \mu_t) + \beta$$

(γ, β: learned parameters)

5. Residual connections

$$f(x) \rightarrow f(x) + x$$

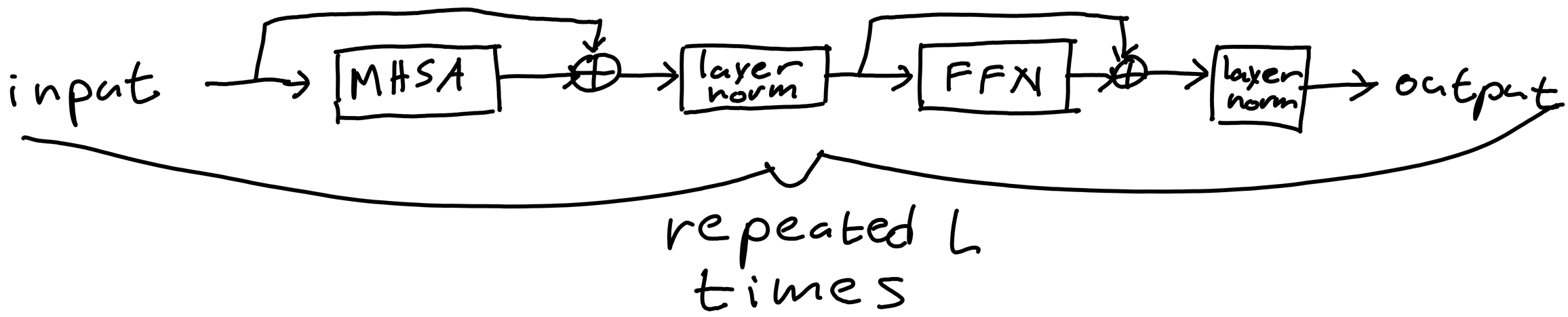
6. "position-wise" feed-forward networks

$$W_2 \, \phi(W_1 x_t + b_1) + b_2$$

↳ nonlinearity, typically ReLU

7. Embeddings

Transformer layer block



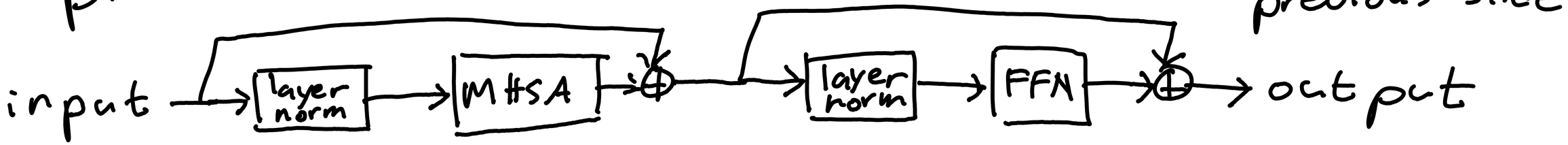
1) Transform using self-attention (mixes over time)
2) Residual connection + layer norm for stability
3) Transform each position independently using FFN

Transformer block modifications

1. Layer norm position (pre- or post-)

↑ diagram from previous slide

pre-LN:



2. Removal of biases

$\underbrace{W_2 \phi(W_1 x_t)}_{\text{FFN}}$ $\underbrace{\frac{g}{\sigma_t}(h_t - \mu_t)}_{\text{RMSNorm}}$

3. FFN nonlinearity → gated linear unit (GLU)

$$W_3(\phi(W_1 x_t) \odot W_2 x_t)$$

↳ often "swish"

$$\phi(x) = x\,\sigma(x)$$

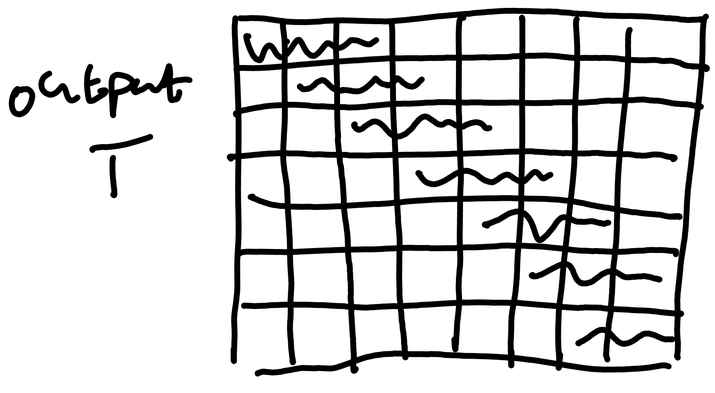
↳ sigmoid

4. Sub-quadratic self-attention

Notes: i) For typical Transformer dimensions and modest sequence lengths ($T < 1000$) compute for FFN $\gg$ compute for MHSA

ii) Can get around quadratic memory cost by being clever (Flash Attention)

e.g. "local" self-attention → only compute attention over a neighborhood around the current position



5. "parallel" block

standard (pre-LN): $x + FFN(LN(x + MHSA(x)))$

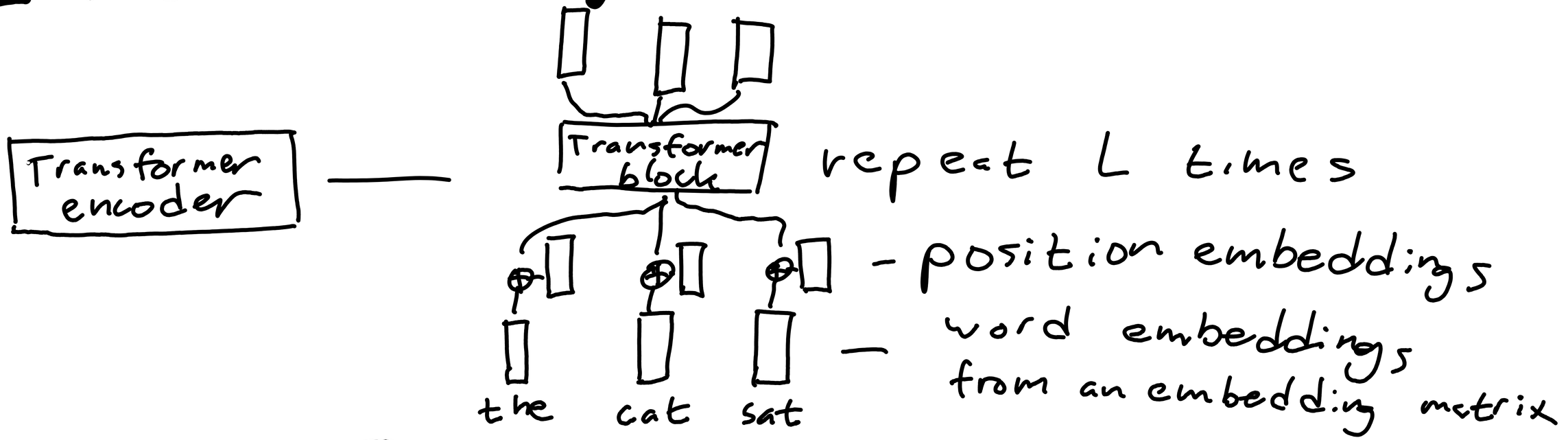
parallel: $x + FFN(LN(x)) + MHSA(LN(x))$

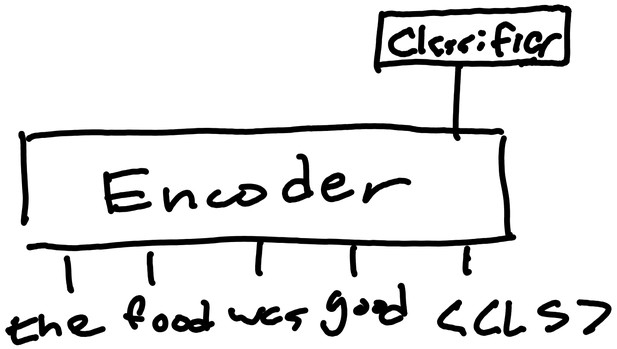
Doesn't harm performance*

Improves GPU utilization

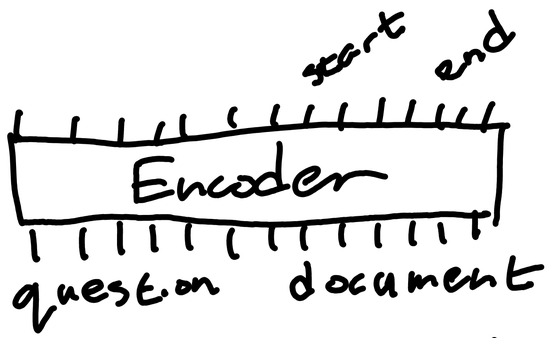
* maybe

Encoder-only Transformer

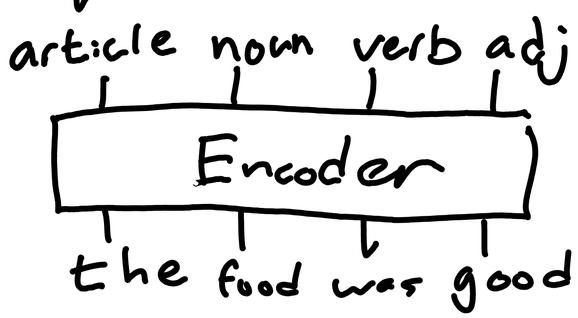




sequence classification



span extraction



sequence labeling

Why pre-train? (first step of transfer learning)

1. Transformer has a limited inductive bias.
   → might help to learn structure before applying the model.
2. Often we can learn structure on a large, data-rich (possibly unsupervised) task.
3. Empirically, pre-training can help
   - improve performance
   - using less labeled data
   - with faster convergence

How to pre-train? (on unlabeled data)

For text, use language modeling!

$$p(x_t \mid x_{t-1}, \ldots, x_1)$$

This only conditions on the past.

So, use "masked" language modeling. (MLM)

$$p(x_t \mid \underbrace{x_T, x_{T-1}, \ldots, x_{t+1}, x_{t-1}, \ldots, x_1}_{\text{entire input sequence other than } x_t})$$

How does BERT implement MLM?

Bidirectional encoder representations from Transformers

the cat sat on the mat

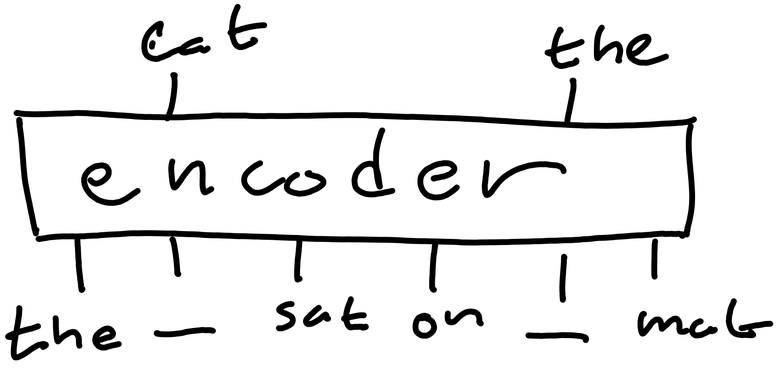
↓

the _ sat on _ mat

$p(x_2 = \text{cat} \mid \text{the } \_ \text{ sat on } \_ \text{ mat})$

$p(x_5 = \text{the} \mid \text{the } \_ \text{ sat on } \_ \text{ mat})$

} pre-train to predict these



basically sequence labelling on the masked-out positions!