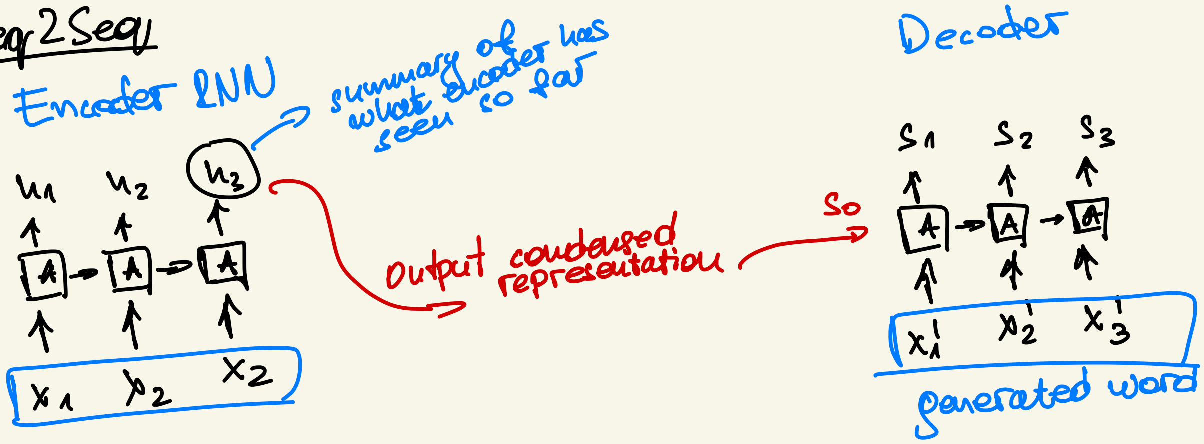
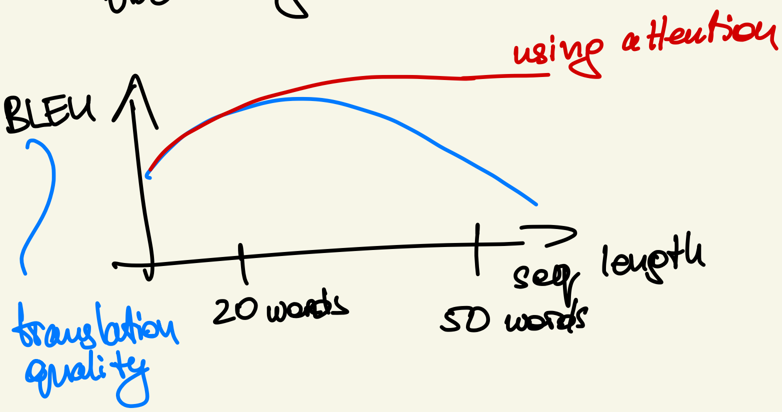
Seq2Seq



each vector x is embedding of a word (english sentence

too many words will lead to forgetting in h_t of encoder



Attention for Seq2Seq

- encoder won't forget
- $O(N^2)$ time complexity

- applied as decoder starts working

From encoder: $h_1 \; h_2 \; h_3 \ldots h_m = s_0$ in decoder

weight $= \alpha_i = \text{align}(h_i, s_0)$ how well h_i and s_0 match 0-1

↑ weight = ↑ relevant

$$\sum_m \alpha_i = 1$$

$$\tilde{\alpha}_i = \underbrace{v^T}_{\text{vector, trainable, shared}} \cdot \tanh\left[\overbrace{W}^{\text{matrix, trainable, shared}} \cdot \begin{pmatrix} h_i \\ s_0 \end{pmatrix}\right]$$

normalize: $\alpha_i = \text{softmax}(\tilde{\alpha}_i)$ so add to 1

- Another align function

$k_i = W_K \cdot h_i$ for $i = 1 \dots m$

$q_0 = W_Q \cdot s_0$

$\tilde{\alpha}_i = k_i^T \cdot q_0$ for $i = 1 \dots m$

$\alpha_i = \text{softmax}(\tilde{\alpha}_i)$

- Context vector

$c_0 = \alpha_1 \cdot h_1 + \dots + \alpha_m \cdot h_m$ weighted avg.

↳ corresponds to s_0

Simple RNN:

$s_1 = \tanh\left(A' \cdot \begin{pmatrix} x_1' \\ s_0 \end{pmatrix} + b\right)$

$x_1' =$ <START> in machine translation

Simple RNN + Attention:

$s_1 = \tanh\left(A' \cdot \begin{pmatrix} x_1' \\ s_0 \\ c_0 \end{pmatrix} + b\right)$

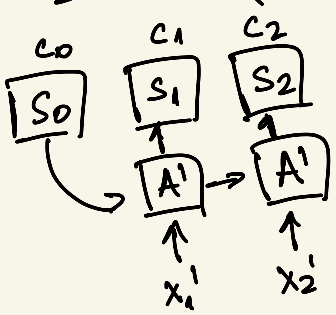
- compute c_1 corresponding to s_1

$\alpha_i = \text{align}(h_i, s_1)$ weights — how well h_i and s_1 are aligned (all encoder hidden states + decoded vector)

- α is recalculated on each run for $s_0 \dots s_n$

$c_1 = \alpha_1 h_1 + \dots + \alpha_m \cdot h_m$

- $s_2 = \tanh\left(A' \cdot \begin{pmatrix} x_2' \\ s_1 \\ c_1 \end{pmatrix} + b\right)$



- complexity: c_j computed from m weights $\alpha_1 \dots \alpha_m$

$m \cdot t$ weights — m = encoder steps, t = decoder steps

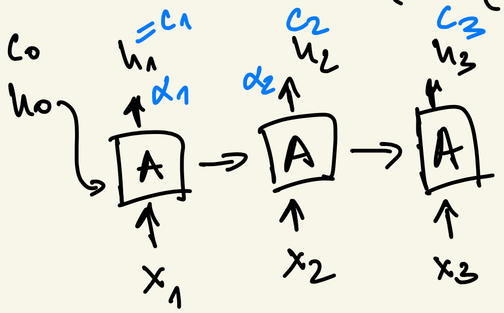
without attention: $O(m+t)$

with: $O(m \cdot t)$

Self Attention for RNN

$$h_1 = \tanh\left(A \cdot \begin{pmatrix} x_1 \\ c_0 \end{pmatrix} + B\right) \qquad h_2 = \tanh\left(A \cdot \begin{pmatrix} x_2 \\ c_1 \end{pmatrix} + B\right)$$

$$h_3 = \tanh\left(A \begin{pmatrix} x_3 \\ c_2 \end{pmatrix} + B\right)$$



Weights: $\alpha_i = \mathrm{align}(h_i, h_2)$ $\quad i$ = each existing state

Ex.: $c_3 = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$

- With self attention RNN is less likely to forget
- Self attention is applied to a single RNN (not Seq2Seq model)
- Pay attention to relevant context in the past

Transformer Model (Attention Layer)

- 2 years after Seq2Seq attention
- 1 year after Self attention
- Seq2Seq, Encoder-Decoder, pure dense layers (no RNN)
- higher accuracy than RNN

- Encoder weights $\quad \alpha_{ij} = \mathrm{align}(h_i, s_j)$ $\quad j$ = decoder hidden state at step j

  for $i = 1 \dots m$ $\quad i$ = encoder state at step i

— calculate align:

- $k_i = W_K \cdot h_i$ $\quad W_K$ = param matrix, random, learns from training data
- $K = [k_1 \dots k_m]$
- $q_j = W_Q \cdot s_j$
- $\alpha_j = \mathrm{softmax}(K^T q_j)$ $\quad K^T$ state vec transformed

$$\alpha_j = \begin{pmatrix} \alpha_{1j} \\ \vdots \\ \alpha_{mj} \end{pmatrix} = \mathrm{Softmax}\left(\begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \end{bmatrix} \underbrace{\begin{bmatrix} \vdots \\ \vdots \end{bmatrix}}_{q_j} \right)$$

- In transformer:
  - q_j = Query (to match others) $= W_Q \cdot s_j$
  - k_i = key (to be matched) $= W_k \cdot h_i$ W to be learnt
  - v_i = Value (to be weighted avg) $= W_v \cdot h_i$

  Context vector: $c_j = \alpha_{1j} \cdot v_1 + \dots + \alpha_{mj} \cdot v_m$

- Intuition

  s_j = decoder output at step j

  - Query : $q_j = W_Q \cdot s_j$
  - key : $k_i = W_k \cdot h_i$ ⤳ for all encoder hidden states: $= K$
  - Weight: $\alpha_j = \text{softmax}(K^T q_j)$
  - Value : $v_i = W_v \cdot h_i$ for all encoder states (m weights)
  - Context vect: $c_j = \alpha_{1j} v_1 + \dots + \alpha_{mj} v_m$

- Attention without RNN
  - Encoder: $x_1 \dots x_m$ english words
  - Decoder: $x'_1 \dots x'_t$ german words
  - start decoder with <start>, continue until receiving <stop>
  - Key $= k_i = W_K \cdot x_i$
  - Value $= v_i = W_V \cdot x_i$

    $$\begin{array}{cccc} k_1 \; v_1 & k_2 \; v_2 & k_3 \; v_3 & \\ \uparrow & \uparrow & \uparrow & \dots \text{ Encoder side} \\ x_1 & x_2 & x_3 & \end{array}$$

  - Query $= q_j = W_Q \cdot x'_j$

    $$\begin{array}{cccc} q_1 & q_2 & q_3 & \\ \uparrow & \uparrow & \uparrow & \dots \text{ Decoder side} \\ x'_1 & x'_2 & x'_3 & \end{array}$$

  - Parameter matrices $W_K / W_V / W_Q$
  - Compute weights: $\alpha_1 = \text{Softmax}(K^T \cdot q_1)$
  - Compute context: $c_1 = \alpha_{11} \cdot v_1 + \dots \alpha_{m1} v_m = V \alpha_1$

- c_j is function of x'_j and $x_1 \cdots x_m$

$c_2 \Rightarrow$ Softmax classifier $\Rightarrow p_2 \rightarrow$ sample German word $\rightarrow x'_3 =$ Embedding of selected word

↳ knows all english + previous german words

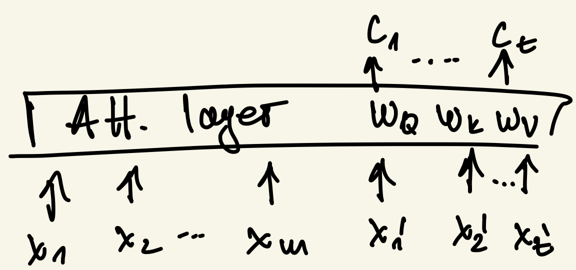
↳ probability distribution over German Vocab

- Attention Layer: $[c_1 \cdots c_t] = \mathrm{Attn}(x, x')$

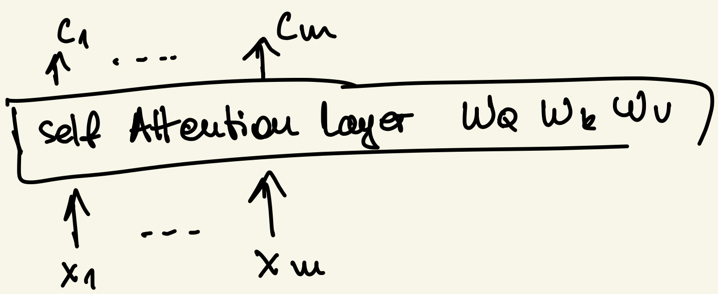
Encoder inputs: $x = x_1 \cdots x_m$

Decoder inputs: $x' = x'_1 \cdots x'_t$

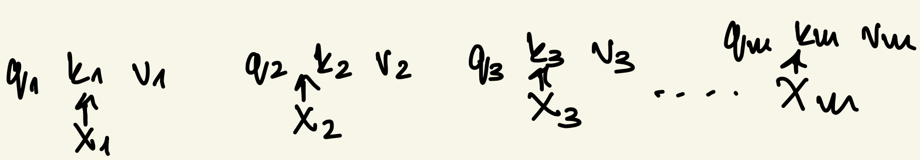
Parameters: W_Q W_K W_V



— Self Attention



$$q_i = W_Q \cdot x_i \quad k_i = W_K \cdot x_i \quad v_i = W_V x_i$$



Weights: $\alpha_{:j} = \mathrm{softmax}(K^T \cdot q_j)$ $\quad j = 1 \dots m$

Context vect: $c_1 = \alpha_{11} \cdot v_1 + \dots + \alpha_{m1} v_m = V \cdot \alpha_1$

$c_j = \alpha_{1j} v_1 + \dots + \alpha_{mj} v_m = V \alpha_{:j}$

$\Rightarrow c_j = V \cdot \mathrm{softmax}(K^T \cdot q_{:j})$

Self attention $[c_1 \cdots c_m] = \mathrm{Attn}(X, X)$ Inputs: $x_1 \cdots x_m$

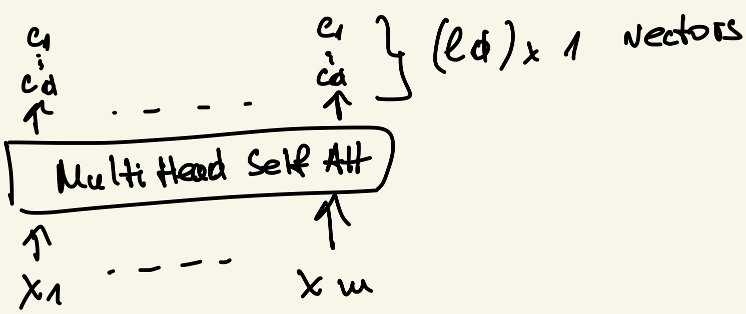
Params: W_Q W_K W_V

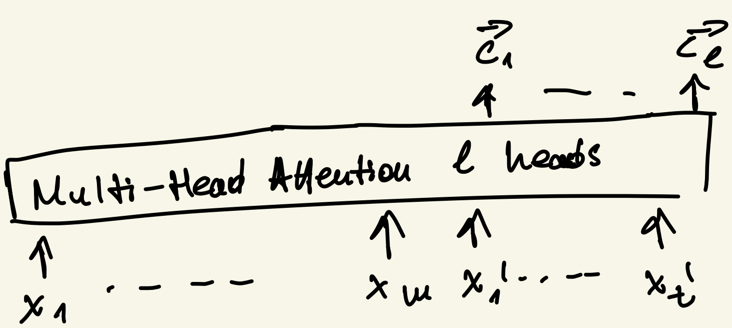
- c_j is influenced by all $x_1 \cdots x_m$
- Attention for Seq2Seq RNN 2015
  Neural machine translation by jointly learning to align and translate
- Self attention for all RNN (not only Seq2Seq) 2016
  LSTM Networks for machine reading
- Attention without RNN 2017
  Attention is all you need

Transformer Model: Deep Neural Network

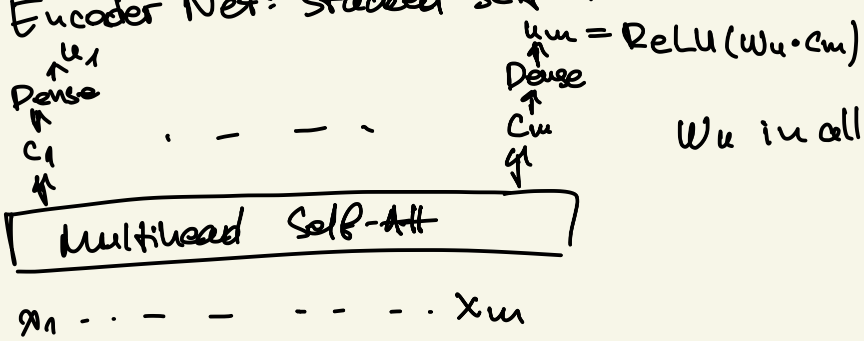
- Attention / self attention = single head attention

— Multihead Self-Attention
  - single head params: W_Q W_K W_V
  - ℓ single head attention (not sharing matrices) has 3ℓ matrices
  - concatenate context outputs of single head



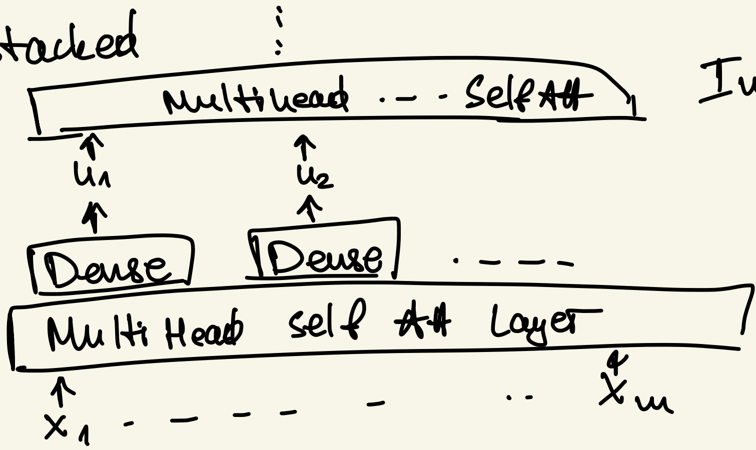


— Encoder Net: Stacked Self-Attention



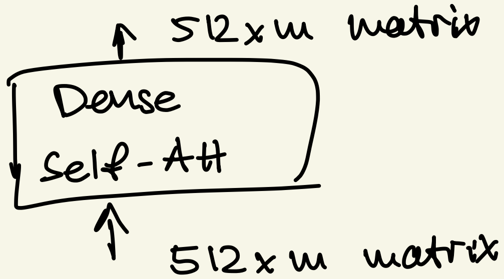
W_u in all Dense layers

- Stacked

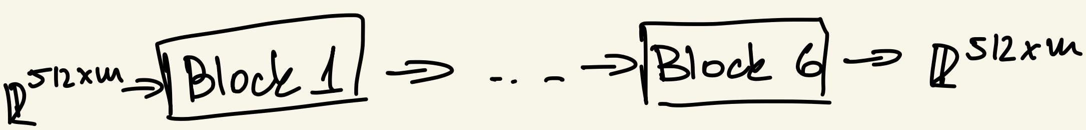


Input = Output len = m

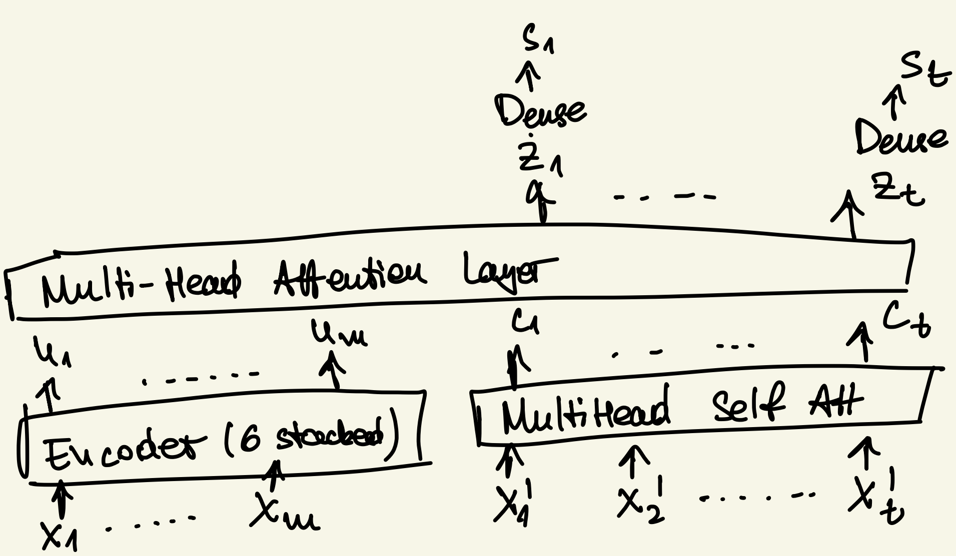
- Block of Encoder



512 can be different (embedding dependent)

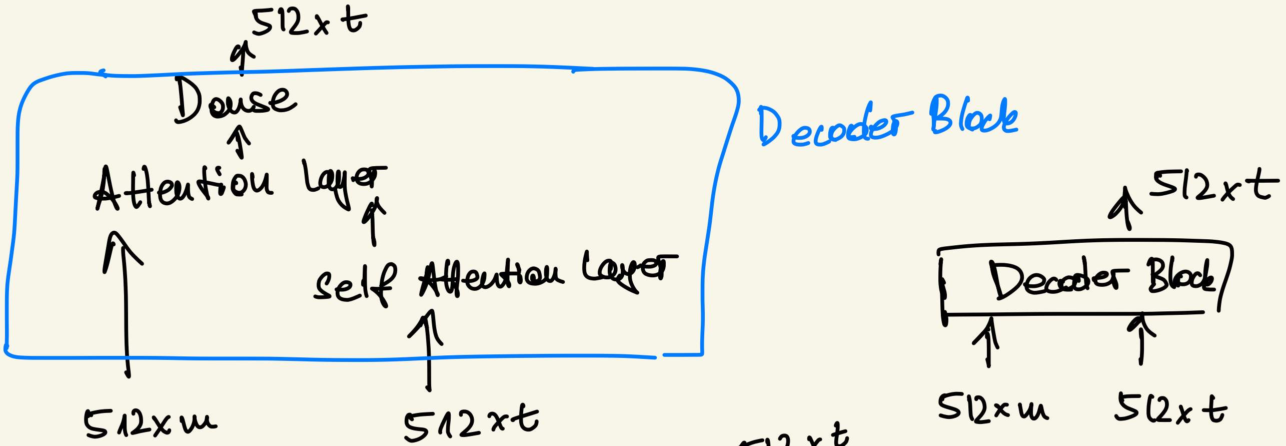


- Decoder Net
  - Transformer is Seq2Seq (encoder + decoder) model
  - Encoder input: embeddings $x_1 \cdots x_m$
  - Decoder inputs: $x_1' \cdots x_t'$ generated t words
  - 1 Block ≈ 1 multi-head attention layer + 1 dense

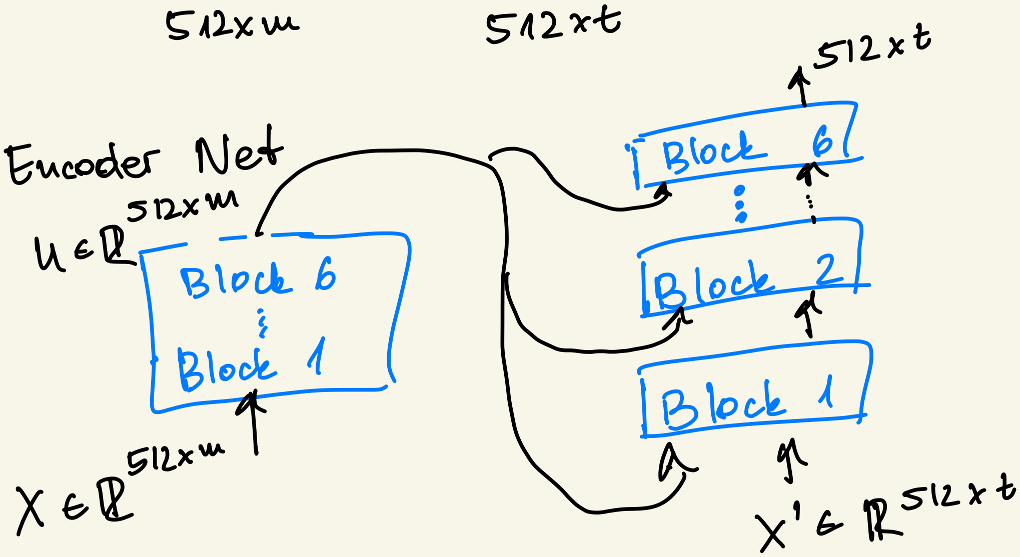


all inputs 512 length

• Decoder Block

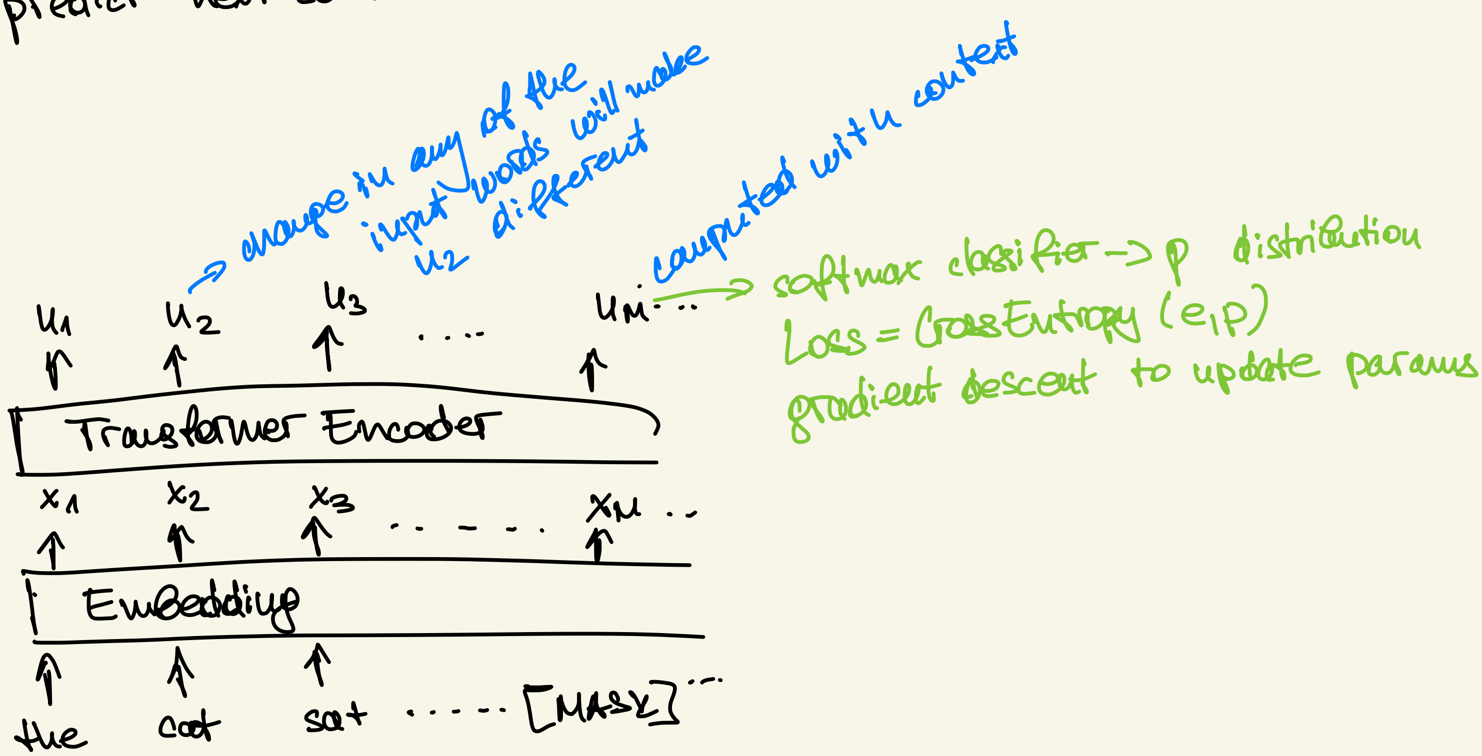


- Encoder Net



Bert for pretrained transformers

- predict masked word
- predict next sentence



– Predict next sentence

- Input [CLS] sentence 1 [SEP] sentence 2
  ↳ for classification; [SEP] ↗ separation

c $u_1 \cdots u_m$ s $h_1 \cdots h_t$ c = 0 not real sentence 2

Encoder
↑
Embedding
↑
[CLS] sentence 1 [SEP] sentence 2

- improves embedding layer
- improves attention on how well words are aligned
- labels are automatically generated

Vision Transformer

image → $\boxed{\text{NN}}$ → $\vec{p}$ n-dimensional for n classes (confidences)

- vision transformer beats CNN 2021
- partition image into patches of same shape
  patch size = 16 x 16 default
  stride = non overlapping patches 16x16 default
- reshape tensors (3 values for RGB) into vectors $d_1 \times d_2 \times d_3$ tensor
  ↳ $x_1 \ldots x_n$ ⇒ $d_1 \cdot d_2 \cdot d_3 \times 1$ vector

$c_0 \to$ softmax → p – n-dim for n classes

c_0 - - - - - - - c_N

skip connection + normalization

encoder:
Dense
MultiHead SelfAtt
Dense
$\boxed{\text{Multi-Head Self-Attention}}$

pos. encoding + $z_1 = Wx_1 + b$ … pos. encoding + $z_N = Wx_N + b$ b/W are shared

z_0 ← embed ← [CLS]

Dense ← x_1 - - - - - - - Dense ← x_N

- Training

Random Initialize → pretrained → FineTuned → Test accuracy

Dataset A → pretrained
very large

Dataset B → FineTuned
smaller for target action

Test set B → Test accuracy

- Datasets

Imagenet small 1,3M
Imagenet med 14M
JFT 300M

ResNet | ViT
100M 300M