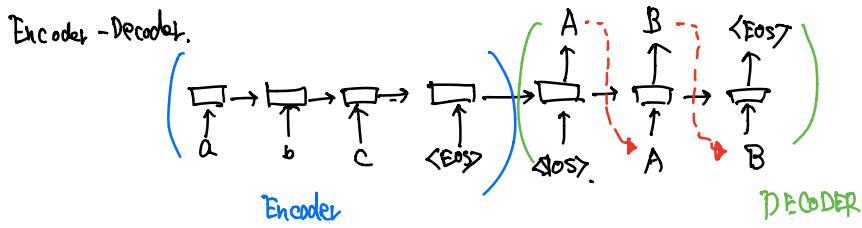


また、attention とは行きたのかを書き出す。



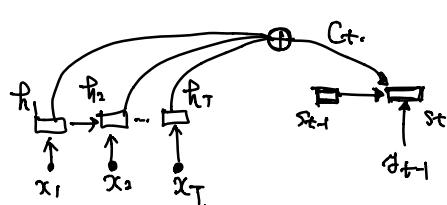
Decoder: 次の最も最後の中間層まで。ここで情報が凝縮されておりだ。

この input のどの部分か重要なかを負けて出力を決めたり。



$$\text{Decoder at time step } t \text{ has } h_t \text{ as } \psi(h_{t-1}, \dots, h_1, \text{enc}) = f(\overbrace{h_{t-1}, \dots, h_1}^{\text{前回の値}}, \underbrace{c_t}_{\text{中間層}}) \text{ method.}$$

中間層  $s_t$  は  $s_t = f(s_{t-1}, h_{t-1}, c_t)$  とかく。



$$s_t = \sum_{i=1}^T a_{ti} h_i \quad \text{各 } a_{ti} \text{ は softmax. } a_{ti} \text{ を乗じて足す.}$$

$$a_{ti} := \frac{\exp(\text{score}(s_t, h_i))}{\sum_{j=1}^T \exp(\text{score}(s_t, h_j))} \quad \text{softmax, } s_t \text{ と } h_i \text{ の score 見る.}$$

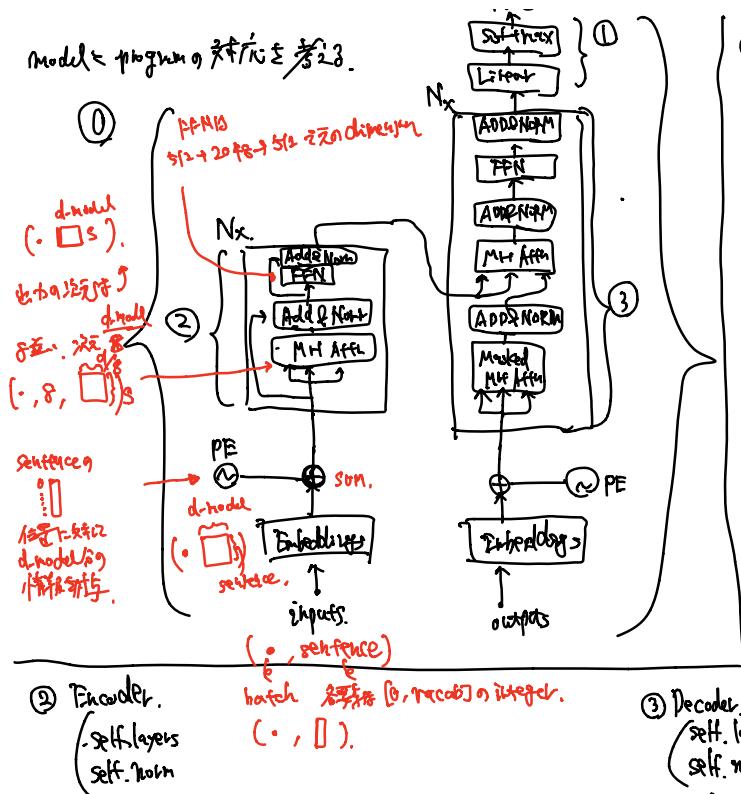
$$\text{score}(s_t, h_i) := \begin{cases} s_t^T W \cdot h_i \\ q_a^T \cdot \text{tanh}(W_1 s_t + W_2 h_i) \end{cases} \quad \text{learnable params}$$

これを全部並列可能。  $s_t = \mathcal{L}(W_s s_{t-1} + W_d d_{t-1}, W_c C_t + b_s)$  ただし  $C_t$  は  $s_t$  の attention によって決まる。 attention は encoder から decoder へ。

ここで、この observation は必ずしも recursive な構造を持つので直角である。



prob.



最終版は normalization で  $\sqrt{2\pi}$  乗算してあります。  
Building block は copy と wrapper です。

## ① 全体を述べる時は

## Encoder Decoder.

self. eur oder  
self. dec oder  
self. grc\_ entw  
self. eng\_ entw  
self. ger entw

~~for~~  $\rightarrow$  for read  $\rightarrow$  detect  
encodes  $\rightarrow$  decode.

generator instance 実装 が、  
後の方で model.generator(..)  
で decide がいつ実行か?

## ① Generatör,

Model의 출력을 토대로, 디코더(decoder) .

基本概念 概念 定义 途中之 Exodus 的 方 向

$$\text{②-1 LayerNorm.} \quad \text{LayerNorm} = \frac{x - \text{mean}}{\text{std}} + B.$$

self.  $a_{-2}$   
 self.  $b_{-2}$   
 self.  $\epsilon_{PS}$

$\text{mean}$   
 $\text{std}$

(2) Sublayer Connection

Sublayer ( $h(x)$ )

$d(x)$

②-3 EncoderLayer.

- self-self-attn
- self-feed-forward
- self-sublayer : Sublayer Connection & ~~pooling~~ (2)
- self-size.

(Add Norm)

formed on instance refind st. すすめ 実施.

③-3 Decoder Layer.

- self. size
- self. self-attch
- self. src-attch : info Encoder  $(M=2)$  to  $T_2 \rightarrow T_3$  activation
- self. feedforward
- self. sublayer

forward refind If self.affn  $\rightarrow$  src- $\text{affn}$   $\rightarrow$  ffn

২২৫- রেফারেন্স  
মুসলিম

`<sublayer[i] (x, lambda x: self.src.cifar10[0])>`

2015年 附帯的の実装(入力)

```
a def affection(g, k, v, mask=None, dropset=None)
```

$$\text{Attention}(Q, k, v) = \text{softmax}\left(\frac{Qk^T}{\sqrt{d_k}}\right)v.$$

(x) 0 लेट

for products.

(\*) ०) ५८.

e class MultiHeadedAffection( ):

- self. of fr
- self. fr
- self. breaks
- self. with
- self. deposit.

〔①②③④〕 名單裏的插入法。  
(nhatch,並列, Separate, d)

1) forward method Ⓛ.

Perhaps  
向量的。  
銅鑄 8 並列  $\frac{5\pi}{8}$  dm

Encoder for 基本特征 input 之  $\hat{x}_t$  为  $\hat{q}, \hat{r}, \hat{v}$  之 合并向量 input !

၃၄၁၂ ရက်  
နေဂတ် (nugget)

$$(\cdot \overbrace{\boxed{\quad}}^{\text{d-model}} \} \text{ sentence.}) \rightarrow (\cdot \overbrace{\boxed{\quad}}^{\text{d-h}} \} \text{ sentence}) \times \text{t}_h (= 8)$$

2)  $\text{affection} \leq \frac{1}{2}$  真用.

3).  $x = x \cdot \text{transpose}(f, 2) \cdot \underline{\text{condjugate}()} \cdot \text{New}(n \cdot \text{hatches}, -1, \text{self. f} \cdot \text{x} \cdot \text{self. d} \cdot \text{e})$

マジカル・ディーラーの  
値をcopy.

## 他の構成要素の理解.

### ① FFN.

$$\text{式} \quad \text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \in \lambda \text{ の } \text{fit} + \lambda \text{ の } \text{fit} + \lambda \text{ の } \text{fit}$$

ただし、 $\lambda$  が大きすぎると、  
学習が止まってしまう。

### ② Embeddings

$$\begin{cases} \text{self.lut} = \text{nn.Embedding}(\text{vocab}, d_{\text{model}}) \\ \text{self.d\_model} \end{cases}$$

基礎として word embedding の特徴を保持し、  
index  $i \geq 20$  の特徴を引いて置き換える。

initialized と初期化  
されたが、なぜか使  
えない。  
PyTorch 説明書  
Initialization.

### ③ Positional Encoding

↓

この公平性は高い！

$$PE = \sum_{d=0}^{d-1} PE_{(pos, d)} = \cos(f(pos, d)) + i \sin(f(pos, d))$$

$$\begin{aligned} PE_{(pos, 2d)} &= \cos(f(pos, d)) \\ PE_{(pos, 2d+1)} &= \sin(f(pos, d)) \end{aligned}$$

pos: pos. (= position).  
model: dim / 1000. 領域.  
正弦波の周期性を用いた  
位置の PE.

↑  
PE:  $\downarrow$  pos. を位  
置 dim: 固定された形  
で固定

↑  
position: pos.  
pos. は固定され  
てある。

## 学習率の実験.

batch.

/self.size

self.srnnak.

padding で埋める部分

batch は同じ new とする。

• batch\_size: enumerate(data\_iter)  
で batch\_size が常に 1 となる。

• batch\_size\_fn: 各 epoch で最大のものさ  
chek して 3 倍を保つでいく。  
→ 3 の倍数。

• 実際に data\_iter はどこで停止する？

$$\left( \text{rebatch}(\text{pred\_size}, b) \text{ for } b \in \overbrace{\text{batch\_itter}}^{\frac{1}{3}} \right)$$
  
MyIterator