(a)

for RNN

Time complexity

time complexity in RNN is directly proportional to square of number of layers. also it is proportional to number of neurons & sequence length. this will be the case in both training & testing time.

$$\boxed{RNN_{Time} \propto l^2 \times n \times t}$$

both train & test

Space Complexity

during testing time it doesnot depend on length of sentences (sequence length).

$$\boxed{RNN_{space\ train} \propto t \times n \times l}$$

$$\boxed{RNN_{space\ test} \propto n \times l}$$

for Transformers.

Time complexity

each embedding in encoder maps to each embedding in decoder. therefore time complexity is proportional to square of sequence length

$$\boxed{Transformers_{time} \propto t^2 \times n \times l}$$

Space complexity

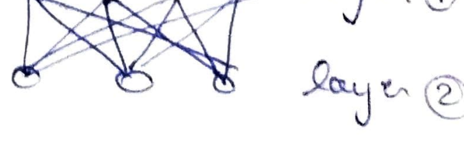
$$\boxed{Transformers_{space} \propto t \times n \times l}$$

1.b) If the number of neurons in each layer is less than sequence length then learning will be effected badly. as same neuron will have to capture the weights that performs for more than 2 word embeddings of sequence

2c) Self attention in encoder layer is not a bottle neck for parallelism as encoder has all the tokens in sequence to compute attention scores.

But for decoder, self attention acts as a bottleneck for parallelism as it depends on the output of previous time step as well.

2d) yes the feed forward network ~~support~~ look across the tokens.



This support parallelism as each layer in feed forward network can be trained on different computational units.

Solution - (2) (a)

The attention vector and weights are defined as :-

$$z = \sum_{i=1}^{m} (v_i \alpha_i)$$

$$\cancel{\alpha_t = \sum_{t=1}^{m}}$$

$$\alpha_i = \frac{\exp(k_i^T q)}{\sum_{1}^{m} \exp(k_i^T q)}$$

- q_i = query vector

  $q_i = W_q x_i$

- $k_i = W_K x_i$ (key vector)
- $V_i = W_V x_i$ (value vector)

$$w_{ij} = q_i^T k_j$$

$$w_{ij} = \text{Softmax}(w_{ij}) = \alpha_i$$

$$\boxed{z_i = \sum_{j} w_{ij} v_j}$$

This means that all the j from 1 to m contributes towards the calculation of z_i.

Solution 2.b

$$\alpha_i = \frac{\exp k_i^T q}{\sum_{i=1}^{m} \exp k_i^T q}$$

- $k_i^T q \approx 0$ for $i \notin \{a, b\}$. (1)

$$z = \frac{v_1 \exp(k_1^T q) + v_2 \exp(k_2^T q) \ldots v_a \exp(k_a^T q) + v_b \exp(k_b^T q) \ldots}{\exp(k_1^T q) \ldots \exp(k_n^T q)}$$

$$\boxed{z \approx \frac{v_a \exp(k_a^T q) + v_b \exp(k_b^T q)}{\exp(k_a^T q) + \exp(k_b^T q)}} \quad (\text{if } n >> 0)$$

\# given in quesion $z \approx \frac{v_a + v_b}{2}$

$$\frac{v_a \exp(k_a^T q) + v_b \exp(k_b^T q)}{\exp(k_a^T q) + \exp(k_b^T q)} = \frac{v_a + v_b}{2}$$

let $\exp(k_a^T q) = a$ & $\exp(k_b^T q) = b$

$$\frac{a.v_a + b.v_b}{a+b} = \frac{v_a + v_b}{2}$$

$$\frac{a v_a}{2} + \frac{b v_b}{2} = \frac{b v_a}{2} + \frac{a v_b}{2} \quad - \text{(11)}$$

from equation 2 we get.

$$a = b$$

$$\exp(Ka^T q) = \exp(Kb^T q)$$

$$\boxed{Ka^T q = Kb^T q} \longrightarrow (3)$$

Solving this analytically we get.

$$\boxed{q = (Ka + Kb).}$$

we can check this by putting in eqn (3)

$$Ka^T(Ka + Kb) = Kb^T(Ka + Kb)$$

$$Ka^T Ka + Ka^T Kb = Kb^T Ka + Kb^T Kb \quad (4)$$

in question it is given $Ki \perp Kj$ also $\|Ki\| = 1$, therefore. eqn 4 can be solved as

$$\underbrace{\cancel{Ka^T Ka}}_{1} + \underbrace{\cancel{Ka^T Kb}}_{0} = \underbrace{\cancel{Kb^T Ka}}_{0} + \underbrace{\cancel{Kb^T Kb}}_{1}$$

$$\boxed{1 = 1}$$

Hence answer to the value of $\boxed{q = Ka + Kb}$

Solution - 3

$$L(q) = \int q(z|x) \cdot \log\left(\frac{p(x,z)}{q(z|x)}\right) dz$$

$$= \int q(z|x) \log\left(\frac{p(z) \cdot p(x|z)}{q(z|x)}\right) dz$$

$$= \int q(z|x)\left(\log\left(\frac{p(z)}{p(z|x)}\right) + \log\left(\frac{p(x|z)}{p(z|x)}\right)\right) dz$$

$$= \int q(z|x)\left(\log\frac{p(z)}{p(z|x)}\right) dz + \int q(z|x)\left(\log\left(\frac{p(x|z)}{p(z|x)}\right)\right) dz$$

$$= -\int q(z|x) \log\left(\frac{p(z|x)}{p(z)}\right) dz$$

$$+ \int q(z|x) \log\left(\frac{p(x|z)}{p(z|x)}\right) \cdot dz$$

$$L(q) = -D_{KL}(q(z|x) \| p(z))$$

$$+ E_{q(z|x)}(\log p(x|z))$$

Solution - 4

to optimise we $\min_p \max_q f(p,q)$

$$f(p,q) = p.q$$

$$\frac{d}{dq} f(q,p) = \frac{d(p.q)}{dq} = p.$$

$$\frac{d}{dp} f(p,q) = \frac{d\, p.q}{dp} = q.$$

Gradient descent.

$\rightarrow q_{t+1} := q_t + \frac{df}{dq}$ {for maximising}

$\rightarrow p_{t+1} := p_t - \frac{df}{dp}$ {for minimising}

$$\boxed{\begin{aligned} &\rightarrow q_{t+1} := q_t + p_t \\ &\rightarrow p_{t+1} := p_t - q_{t+1} \end{aligned}}$$

	t	0	1	2	3	4	5	6
$(p_t+q_t) \leftarrow$	q	1	2	1	-1	-2	-1	1
$(p_t - q_{t+1}) \leftarrow$	p	1	-1	-2	-1	1	2	1

(b) By using above approch it is not possible to reach the optimal for all the parameters as we see in above example at $t=6$ the values of q & p both reaches back to where it started.

(c) In GAN, we do not simultaneously reach the optimal for all parameters. instead we achieve Nash equillibrium where each parameter can not reduce their cost without altering the cost of other parameters.