1 Question 1

Suppose that $K_x, Q_x, V_x \in \mathbb{R}^{n_x \times d}$ are respectively the key, query and value matrices of the sequence x, where n_x is the length of x (each token w of the sequence is represented by 3 vectors key k_w, query q_w and value v_w in $\mathbb{R}^d$ which are the corresponding rows of the matrices). The matrices K_y, Q_y, V_y are defined similarly. Then:

- In the case of attention of y over x, the attention is computed as

$$\text{attention}(y|x) = \text{softmax}\left(\frac{Q_y K_x^\top}{\sqrt{d}}\right) V_x \in \mathbb{R}^{n_y \times d} \tag{1}$$

- In the case of self-attention over x, the attention is calculated as

$$\text{self-attention}(x) = \text{softmax}\left(\frac{Q_x K_x^\top}{\sqrt{d}}\right) V_x \in \mathbb{R}^{n_x \times d} \tag{2}$$

2 Question 2

1. Let us first consider the **Self-Attention** framework.

- **Complexity**: Looking at the equation (2), we see that: computing $\left(QK^\top\right)$ takes $O(n \times d \times n) = O(n^2 d)$ time, then computing $\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)$ take $O(n^2)$ time and computing $\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V$ takes $O(n^2 d)$ time. In total the complexity is $O(n^2 d)$.
- **Sequential Operations**: As we can see in the equation (2), the input of each word t is independent of the output of other words, so the complexity of sequential operations is $O(1)$.
- **Maximum Path Length**: As we can see in the self-attention architecture, every pair of words in the sentence is connected in constant time (by the dot product of key and query vector). Hence, the maximum path length of a self-attention layer is $O(1)$.

2. Now we consider a **Recurrent** layer. We first recall the formula of a vanilla recurrent layer over a sequence $x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(n)}\right)$, with $x^{(t)} \in \mathbb{R}^d$

$$h^{(t)} = g\left(W_{hh} h^{(t-1)} + W_{hx} x^{(t)} + b_h\right) \quad \text{for } t = 1, \ldots, n \tag{3}$$

where $h^{(t)} \in \mathbb{R}^d$ is the hidden representation at time t, $W_{hh}, W_{hx} \in \mathbb{R}^{d \times d}$ and $b_h \in \mathbb{R}^d$ are the weights of the model (here we assume for simplicity that the dimensionality of the input vectors is equal to that of the hidden vectors) and g is an activation function.

- **Complexity**: We consider each time step t, computing $(W_{hh} h^{(t-1)}$ and $W_{hx} x^{(t)}$ both take $O(d \times d \times 1) = O(d^2)$, then computing $g\left(W_{hh} h^{(t-1)} + W_{hx} x^{(t)} + b_h\right)$ takes $O(d^2)$ times. Hence the complexity for one step is $O(d^2)$, thus results in an $O(nd^2)$ complexity in total. Note that the complexity still applies to other variants of the vanilla RNN such as LSTM, GRU or bi-directional RNN where we only add up a constant number of $O(nd^2)$ operations.
- **Sequential Operations**: The recurrent layer takes the hidden output of the previous time step as an input for the current time step, hence the number of operations that need to be computed in a sequential order for a sequence of n words is $O(n)$.
- **Maximum Path Length**: In a recurrent layer, only two consecutive words are connected. Hence the length between the first word and the last word in the sentence is $O(n)$.

3. We finally consider a CNN layer, where the output $\mathbf{o} = (o_1, \ldots, o_n) \in \mathbb{R}^n$ of a filter $\mathbf{F} \in \mathbb{R}^{k \times d}$ over a sequence $x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(n)}\right)$, with $x^{(t)} \in \mathbb{R}^d$ is computed as follows

$$o_t = f\left(\mathbf{F} \cdot \mathbf{X}_t + b_{\mathbf{F}}\right) + b \quad \text{for } t = 1, \ldots, n \tag{4}$$

where $\mathbf{X}_t = x^{(t-\lfloor k/2 \rfloor : t+\lceil k/2 \rceil - 1)} \in \mathbb{R}^{k \times d}$ (we assume using a zero padding to preserve the dimension of the output, hence $x^{(t)} = 0$ for $t \leq 0$ or $t \geq n+1$), $b_{\mathbf{F}}$ and $b \in \mathbb{R}$ are biases, f is an activation function and the $\cdot$ denotes the sum of element-wise multiplication of the two matrices.

- **Complexity**: We see that for each filter $\mathbf{F}$, computing $\mathbf{F} \cdot \mathbf{X}_t + b_{\mathbf{F}}$ takes $O(kd)$ time, then computing $f\left(\mathbf{F} \cdot \mathbf{X}_t + b_{\mathbf{F}}\right) + b$ takes $O(kd)$ time as well. Summing over n time steps, we deduce that the complexity for a filter is $O(knd)$. However, in order to produce a d-dimensional output for each word, we need d such filters, resulting in a complexity of $O(knd^2)$ in total.
- **Sequential Operations**: The convolution layers does not require any previous outputs to compute the output of the current time step. Hence the sequential complexity is $O(1)$.
- **Maximum Path Length**: In a convolution layer, every k consecutive words are connected. Hence the length between the first word and the last word in the sentence is $O(n/k)$.

3 Question 3

Using multiple heads of small dimension allows the model to

- Capture more attentional properties between words of the sequences, for example one head may count for the possessive relations of the words while other head can contain the relation between subject and object in the sentences.
- On the other hand, reducing the dimension of the heads permits to remain the same number of parameters as for one big attention head. This is because the complexity of self-attetion layer is $O(n^2 d)$, hence using h heads of dimension d equals to using one head of dimension hd.

4 Question 4

We have the formula of the positional encoding

$$\begin{aligned} PE(pos) &= \left(\sin\left(\alpha_0 pos\right), \cos\left(\alpha_0 pos\right), \ldots, \sin\left(\alpha_M pos\right), \cos\left(\alpha_M pos\right)\right)^\top \\ PE(pos+k) &= \left(\sin\left(\alpha_0 (pos+k)\right), \cos\left(\alpha_0 (pos+k)\right), \ldots, \sin\left(\alpha_M (pos+k)\right), \cos\left(\alpha_M (pos+k)\right)\right)^\top \end{aligned} \tag{5}$$

where $\alpha_i = \dfrac{1}{10000^{2i/d_{model}}}$, and $2M = d_{model}$ is the dimensionality of word representations. Since we have

$$\begin{aligned} \sin\left(\alpha_i (pos+k)\right) &= \sin\left(\alpha_i pos\right)\cos\left(\alpha_i k\right) + \cos\left(\alpha_i pos\right)\sin\left(\alpha_i k\right) \\ \cos\left(\alpha_i (pos+k)\right) &= \cos\left(\alpha_i pos\right)\cos\left(\alpha_i k\right) - \sin\left(\alpha_i pos\right)\sin\left(\alpha_i k\right) \end{aligned} \quad \forall i = 0, 1, \ldots, M \tag{6}$$

We then have

$$\begin{bmatrix} \sin\left(\alpha_0 (pos+k)\right) \\ \cos\left(\alpha_0 (pos+k)\right) \\ \vdots \\ \sin\left(\alpha_M (pos+k)\right) \\ \cos\left(\alpha_M (pos+k)\right) \end{bmatrix} = \begin{bmatrix} \cos\left(\alpha_0 k\right) & \sin\left(\alpha_0 k\right) & 0 & \ldots & 0 \\ -\sin\left(\alpha_0 k\right) & \cos\left(\alpha_0 k\right) & 0 & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \ldots & 0 & \cos\left(\alpha_M k\right) & \sin\left(\alpha_M k\right) \\ 0 & \ldots & 0 & -\sin\left(\alpha_M k\right) & \cos\left(\alpha_M k\right) \end{bmatrix} \begin{bmatrix} \sin\left(\alpha_0 pos\right) \\ \cos\left(\alpha_0 pos\right) \\ \vdots \\ \sin\left(\alpha_M pos\right) \\ \cos\left(\alpha_M pos\right) \end{bmatrix} \tag{7}$$

which shows the linear property of $PE(pos)$.
This positional encoding is suitable encoding because

- The values for different positions are distinctive enough to be distinguished, thus resulting in a good map from position to encoding vectors.
- The fact of alternating sin and cos functions constantly allows the encoding vectors to be more constrastive.

5 Question 5

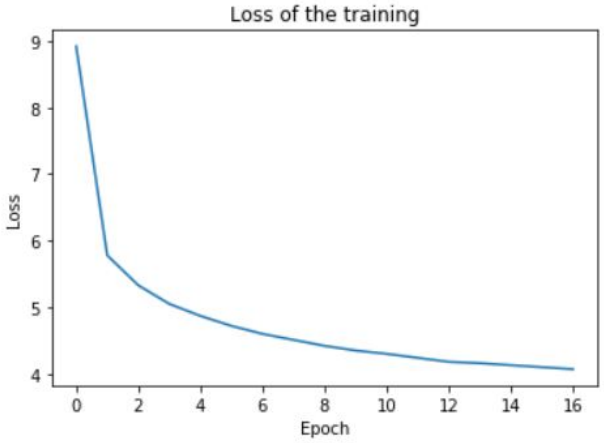
By masking out the words to the right of the word at a given step, the model only uses the information of the previous translated words, as well as the encoding vector of the source sequence to generate the desired word. Hence it can prevent using the forwards information to predict the current word.

6 Question 6

As shown in [2], the number of parameters for a base Transformer model is roughly 65M. This mostly comes from embedding variables ($d_{model} \times n_{source} + d_{model} \times n_{target}$), the parameters of feed-forward layers after self-attention ($2 \times N_x \times d_{model} \times d_{ff}$), projections matrices for the attentions ($3 \times N_x \times d_{model} \times d_{model}$), and the final feed-forward layer to output ($d_{model} \times n_{target}$). Therefore, it's nearly impossible to train the model even with a GPU. The authors claimed to train the model on 8 NVIDIA P100 GPUs, and it took 12 hours to train 100,000 steps and achieve a reasonable result.

7 Question 7

We success to train the model for the first 17 epochs and receive the following loss history.



However, the model then start to expose and not decrease anymore. We found that with different initialization, the models achieve to different loss, this might be due to the fact that we do not use some generalization methods such as dropout or normalisation. This may also come from some errors in the implementation of the model. The resulting sentences are therefore not good, but it still captures some information. For example:

- I am a student $\rightarrow$ suis suis ... suis
- I did not mean to hurt you $\rightarrow$ mal mal ... mal
- The fridge is full of food $\rightarrow$ nourriture nourriture ... nourriture

If we further force the model to predict only non overlaping words, we can see that the model can already handle the information of the source sentence, and is able to generate some meaningful phrases. For example, we show here also the tokens after the $< EOS >$ token to the the ability of the model:

- I am a student $\rightarrow$ ['$< SOS >$', '$< EOS >$', '.', 'un', 'je', 'suis', 'étudiant', 'étudiante',...]
- The river is full of fish $\rightarrow$ ['$< SOS >$', '$< EOS >$', 'le', '.', 'est', 'de', 'la', 'rempli', 'rivière', 'très', 'ce', 'plein',...]
- I can't help but smoking weed $\rightarrow$ ['$< SOS >$', 'de', '$< EOS >$', '.', 'pas', 'fumer', 'n', 'ne', 'ai', 'd', 'l', 'je', 'à', 'arrêter',...]

8 Question 8

We present here the ideas presented in [1]. It introduce BERT (Bidirectional Encoder Representations from Transformers) which aims to use a multi-layer bidirectional Transformer encoder to obtain the representation vectors of sentences by training on sequences of sentences in a given language.
More formally, let $\mathcal{D}$ be the set of all the sentences (can be a series of contiguous text, not necessarily a true sentence) in a given language. Then the input for the BERT can be one of the forms:

$$[CLS], x^{(1)}, x^{(2)}, \ldots, x^{(n_x)} \quad \text{or} \quad [CLS], x^{(1)}, x^{(2)}, \ldots, x^{(n)}, [SEP], y^{(1)}, y^{(2)}, \ldots, y^{(n_y)}$$

where $x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(n_x)}\right)$ and $y = \left(y^{(1)}, y^{(2)}, \ldots, y^{(n_y)}\right) \in \mathcal{D}$, $[SEP]$ is a special separation token used to separate the two sentences in the sequence, and $[CLS]$ is a special token for classification task, which is used

tell if the sentence y really precedes the sentence x in reality. The objective of using 2 sentences at the input here is to help the model to perform down-stream tasks such as Question Answering or Sentence Prediction. In order for the model to learn effectively the embedding of the sequences, they used 2 different unsupervised tasks for training:

1. **Masked Language Model.** In order for the model to learn the representation in both directions, they apply a mask with a probability p at each words of the input, then force the model to predict these words. For example:

$$[CLS], I, wake, [mask1], this, [mask2], [SEP], Then, I, [mask3], to, school, [SEP]$$
$$\rightarrow [mask1] : up, [mask2] : morning, [mask3] : go$$

2. **Next Sentence Prediction.** To learn the relationship between sentences, they use the output value of the token [CLS] to indicate wether the sentences y follows the sentence y. For example:

$$[CLS], I, am, tired, [SEP], I, need, to, sleep, [SEP]$$
$$\rightarrow [CLS] : IsNext$$
$$[CLS], I, am, tired, [SEP], The, child, take, the, ball, [SEP]$$
$$\rightarrow [CLS] : NotNext$$

Note that the output representations of the $[mask]$ and $[CLS]$ tokens are used to predict the corresponding words. The input vectors of these tokens are obtained by summing the embedding vectors with the corresponding position embeddings as well as segment embeddings (indicating if the token belong the first or the second sentence). This helps the model to better identify the two sentences.
This model is indeed very efficient as it takes the advantages of the self-attention structure of the Transformer model. As multiple heads are used, the model manages to capture many of the inner properties of the sentences, hence leads to very good performances of the model.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.