Lab5 Image Captioning with Transformer

Outline

In this lab, you need to implement transformer decoder layer and multi-head attention. Throughout this lab, you will understand training details and architecture of the transformer model.

Deadline

- Report: 2022/1/19
- Demo: 2022/1/19
- email title: DLP_Lab5_your student id_name
- email to hibb@iis.sinica.edu.tw

5.1 Requirements

- implement forward function in transformer decoder layer.
- implement multi-head attention forward function.
- train your transformer model on the MSCOCO dataset.
- Run sample.py with your own image.

5.1.1 Required knowledge

1. How to calculate attention weights
2. How does transformer decoder propagate information

5.2 Demo

- Run sample.py with your own image.
- one or two questions related to transformer.

5.3 Report Spec

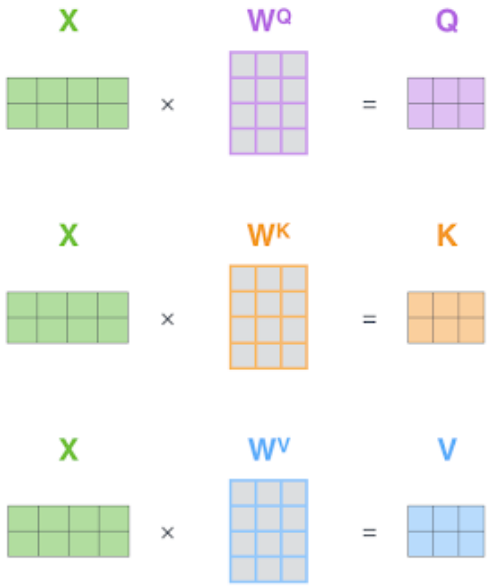
1. Introduction (5%)
2. Explain how you implement transformer (45%)
3. Output result from sample.py (20%)
4. Discussion (10%)
5. Demo (20%)

5.4 Details

5.4.1 Attention

How do we obtain Q, K, V?
It is a linear transformation to the input. If we generate Q, K, V with the same input x, then we are performing self-attention.

How about cross-attention?
Generate K, V with another target, for example, in a Transformer block, we perform cross attention with decoder input and encoder output after the self-attention.



Every row in the X matrix corresponds to a word in the input sentence. We again see the difference in size of the embedding vector (512, or 4 boxes in the figure), and the q/k/v vectors (64, or 3 boxes in the figure)

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

scaled

5.4.2 Multi-head attention

Basically, multi-head attention performs the same as attention. But we separate dimensions into small chunks.
For example, if we have the hidden dimension of 100, and head amount of 5, we compute attention on every $100/5 = 20$ dimensions.
How do we separate dimensions? A useful function `torch.reshape`

```
import torch

x = torch.rand(64, 50, 100)
# (batch size, sequence length, hidden dimension)
x = x.view(64, -1, 5, 20).permute(0, 2, 1, 3)
# (batch size, head size, sequence length, head dimension)
```