Data C182 Designing, Visualizing & Understanding DNN
Fall 2024 Eric Kim, Naveen Ashish

Discussion 06

This discussion covers self-attention mechanism, multi-head attention, encoder architecture, attention vs CNN, and Cross attention.

1. Attention Mechanisms

For many NLP and visual tasks we train our deep models on, features appear on the input text/visual data often contributes unevenly to the output task. For example, in a translation task, not the entirety of the input sentence will be useful (and may even be confusing) for the model to generate a certain output word, or not the entirety of the image contributes to a certain sentence generated in the caption.

While some RNN architectures we previously covered possess the capability to maintain a memory of the previous inputs/outputs, to compute output and to modify the memory accordingly, these memory states need to encompass information of many previous states, which can be difficult especially when performing tasks with long-term dependencies.

Attention mechanisms were developed to improve the network's capability of orienting perception onto parts of the data, and to allow random access to the memory of processing previous inputs. In the context of RNNs, attention mechanisms allow networks to not only utilize the current hidden state, but also the hidden states of the network computed in previous time steps as shown in the figure below.

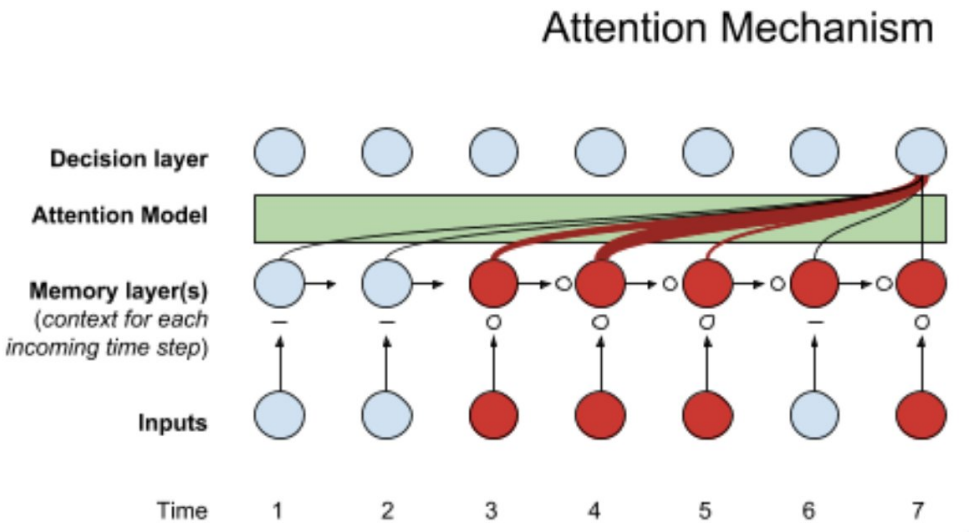

Figure 1: Attention mechanism

0.1 Self-Attention in Transformer Networks

Self attention is an attention mechanism introduced in the Transformer architecture. The first step of the attention is to compute Q, K, V using different transformations from the original input embedding as shown in the figure below.

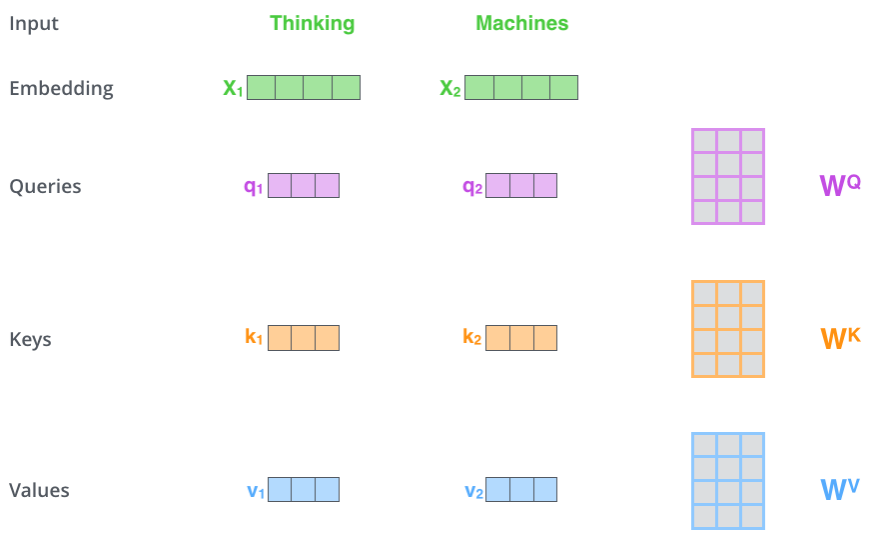

Figure 2: Computing K, Q, V from input embeddings in a Transformer Network

Then, using Q and K, we can compute a dot product as the 'score' of K for Q as shown in the figure below. Intuitively, Q is the querying term that you would like to find. Its relations for each corresponding K and V pairs (key-value) pairs, can be computed using the key. Note that this dot product is computed across various time steps by matrix multiplication. So we get a score for each K for each Q. We then use a Softmax function to get our attention weights.

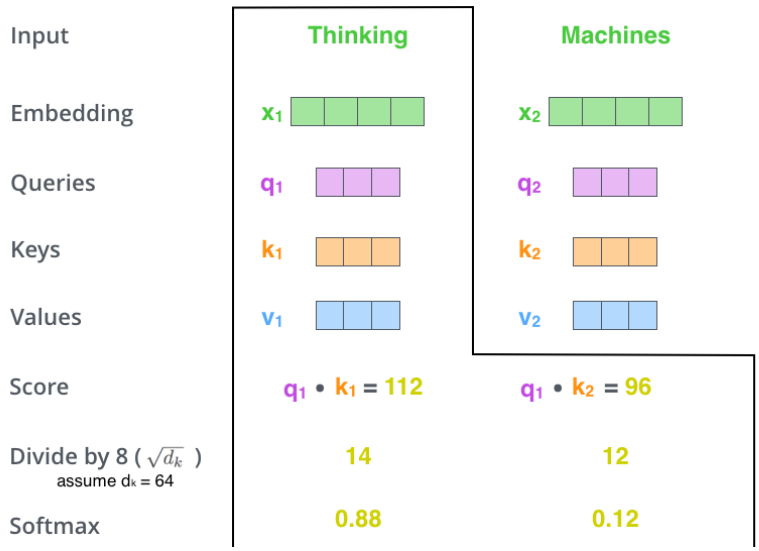

Figure 3: Computing Attention Scores from K, Q, V

Finally, using these weights, we can compute our weighted sum by multiplying the weights with the values. You may find the follow figure helpful.

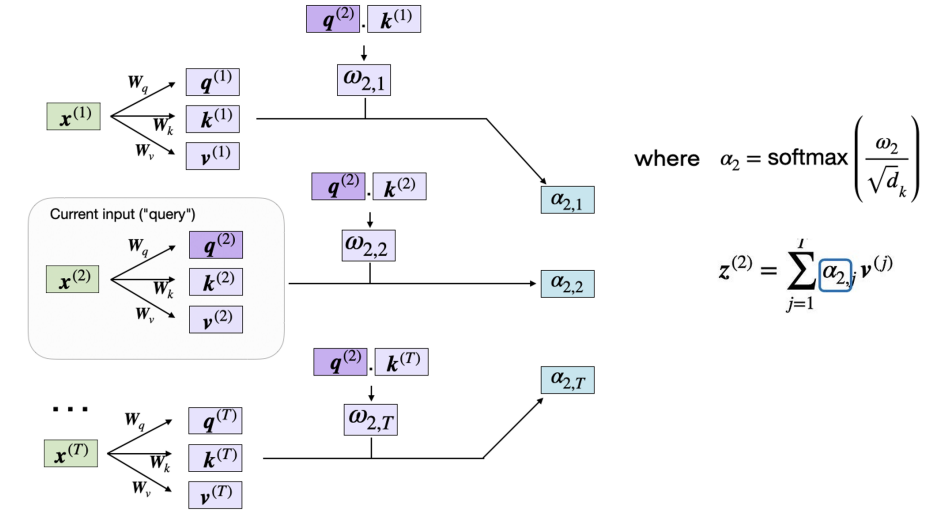

Figure 4: Attention operation

(a)

Problem: Generalized Attention in Matrix Form

Consider a form of attention that matches query q to keys $k_1, \ldots, k_t$ in order to attend over associated values $v_1, \ldots, v_t$.
If we have multiple queries $q_1, \ldots, q_t$, how can we write this version of attention in matrix notation?

(b)

Problem: Justifying Scaled Self-Attention

In practice, Transformers use a Scaled Self-Attention. Suppose $q, k \in \mathbb{R}^d$ are two random vectors with $q, k \sim \mathcal{N}(\mu, \sigma^{\in}\mathcal{I})$, where $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^+$

i. Define $\mathbb{E}[q^\top k]$ in terms of μ, σ, d

ii. Define $Var(q^\top k)$ in terms of μ, σ, d

iii. How does $Var(q^\top k)$ scale with d? Why might this be problematic?

iv. Suppose we scale the dot product by $\frac{1}{s}$, i.e. $Var(q^\top k/s)$. What value of s should we choose to address the issue from 3.?

Problem: Non-local Means Interpretation of Attention

Given a dataset of $(x, y)_i$, a non-parametric method for estimating the value for arbitrary x is via "averaging the y of the nearest data points to x.

$$y = \sum_i \frac{K(x, x_i)}{\sum_j K(x, x_j)} y_i$$

Where $K(x, x_i)$ is the "similarity" between x and x_i, and y is computed as the weighted average. How does this relate to transformer, specifically, the q, k, v matrix?

0.2 Multi-Headed Attention

The multi-head self-attention module is a key component in Transformer. Rather than only computing the attention once, the multi-head mechanism splits the inputs into smaller chunks and then computes the scaled dot-product attention over each subspace in parallel. The independent attention outputs are simply concatenated and linearly transformed into expected dimensions.

$$
\begin{aligned}
\mathbf{X}_0 &= \mathbf{X}[:, : d_k] \\
\mathbf{X}_1 &= \mathbf{X}[:, d_k : 2d_k] \\
\mathbf{X}_2 &= \mathbf{X}[:, 2d_k : 3d_k] \\
&\ldots\ldots \\
\text{head}_i &= \text{Attention}(\mathbf{X}_i\mathbf{W}_i^q, \mathbf{X}_i\mathbf{W}_i^k, \mathbf{X}_i\mathbf{W}_i^v) \\
\text{MHA}(\mathbf{X}, \mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v) &= \text{Concat}[\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^o
\end{aligned}
$$

Here X_i is the i-th split of size d_k along the embed dimension where $d_k = \frac{d}{h}$.

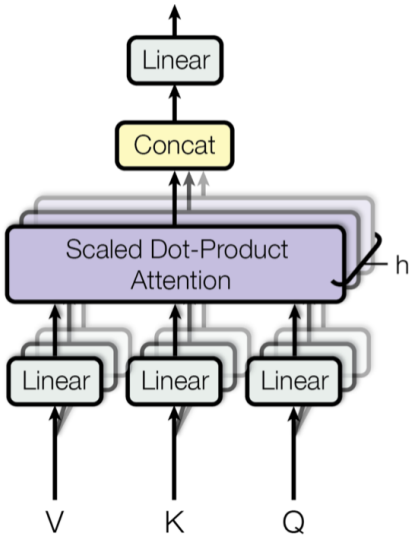

Figure 5: Multi-headed attention

Problem: Properties of Multi-head attention

i. i. We want to change single-head attention to multi-head attention. The original single-head attention has hidden dimension of d, d_k keys, d_q queries and d_v values. What is the relationship between d_k and d_q?

ii. Suppose the sequential input that the user feeds in is of length n What is the total computational cost of self attention operation? Only consider matrix multiplications. Assume the computation is the product of the number of elements in the two matrices. You may find it helpful to walk through the following diagram.

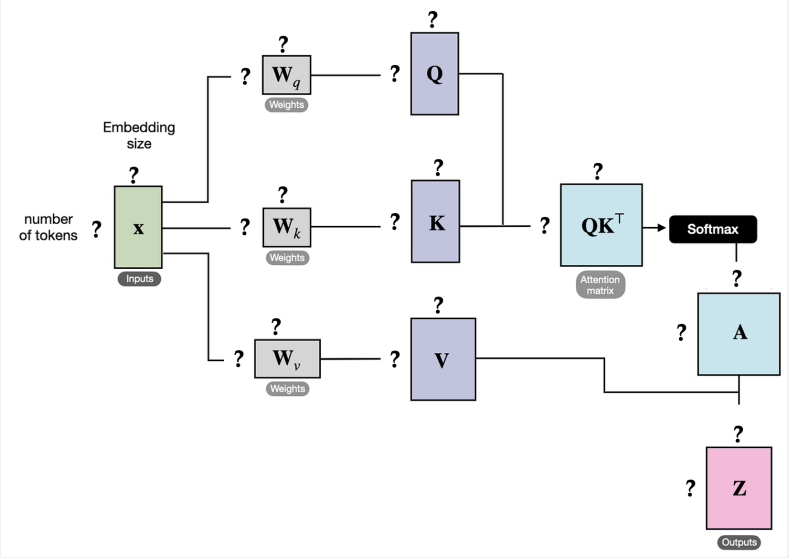

Figure 6: Self attention

iii. Now we want to maintain the same number of parameters, and use h heads. How many keys, queries and values will there be? What is the computational advantage of using MultiheadAttention?

(c) Encoder Architecture

Among the famous transformer-based architectures, encoder-only transformers are popular for image classification tasks. Below are from the famous BERT and ViT paper

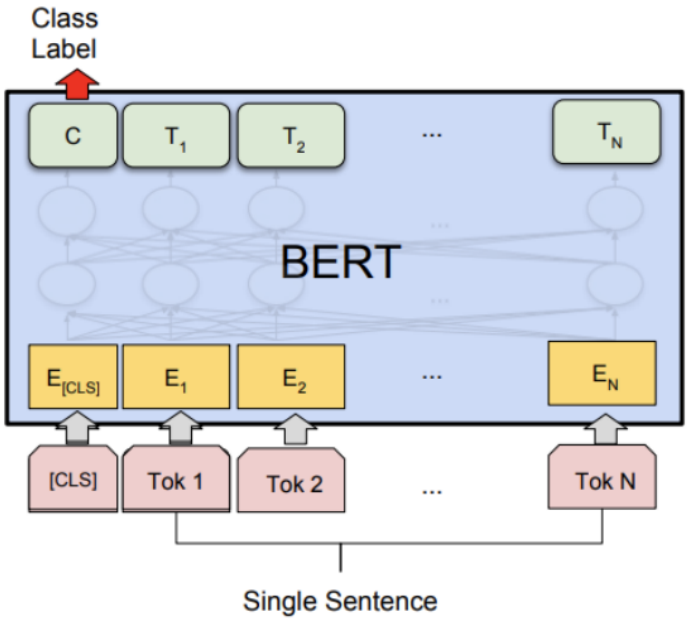

Figure 7: Encoder: BERT

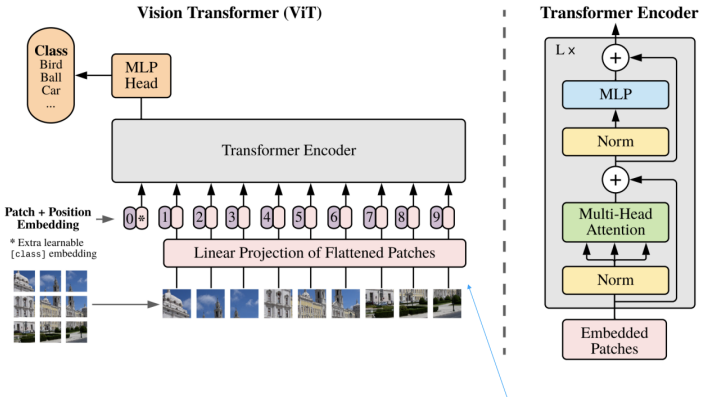

Figure 8: Encoder: ViT

Encoder Architecture

(d) i. Explain how classification works for encoder-only architectures

ii. How does transformer-based encoder compare to CNN?

(e) Cross Attention

What is cross-attention, and how does it differ from self-attention?

In self-attention, we work with the same input sequence. In cross-attention, we mix or combine two different input sequences. Note that in cross-attention, the two input sequences x_1 and x_2 can have different numbers of elements. However, their embedding/hidden dimensions must match.

Cross attention is useful for fusing together different sources of information, for instance, combining text and image data, or in autonomous driving, combining vehicle information with the roadgraph

information. In machine translation, one can also combine sentences of two different language with cross attention.

Cross Attention

i. i. Fill in the dimension in the diagram.

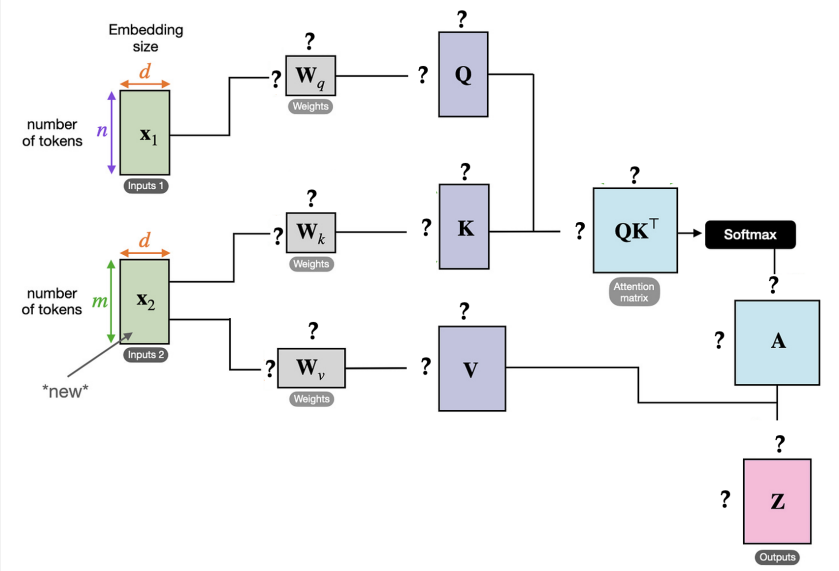

Figure 9: Cross Attention mechanism

