Train Visual Models from Texts Supervision

Zhe Li

In this note, we will summarize how image, text is encoded and how image and text are jointly used to train vision-language model in this seminal paper [1]. Specifically, we use the model ViT-B/32 model as the example. In the Section 1, we illustrate how image is encoded to an embedding vector. In the Section 2, we illustrate how text is encoded to an embedding vector. In the Section 3, we explain how image and text embedding features are jointly used to train vision-language models.

1 Image Encoder

In the Fig 1, we illustrate how to encode an image to an embedding vector. Assume that the input image is the size $3 * 224 * 224$. Firstly, we conduct 2D-Conv with kernel size $32 * 32$ and stride $(32, 32)$, thus in each output channel the input image is converted to $\frac{224}{32} * \frac{224}{32} = 7 * 7 = 49$ numbers, the predefined number of output channel in this layer is 768. This step is so-called patchfy.

We concatenate one extra class embedding vector with size $1 * 768$ to the output of patchfy operation, then we have $50 * 768$ matrix, specifically we pad the class embedding vector with size $1 * 768$ as the first row in the $50 * 768$ matrix. In the final step, we will see why we need to mention this.

To encode each patch's positional information, there is the positional embedding matrix with size $50 * 768$, each row is representing one position. We have output matrix with size $50 * 768$ after adding positional embedding matrix.

Inside the red dash rectangle is attention block. In this specific ViT-B/32 model, there are 12 repeated attention blocks. There are layer normalization layer, multi-head attention layer, fully connected layers, GeLu layers within attention block. For the detail of the attention layer, we summarize it in other notes. The first fully connected layer projects the 768 low dimension to the 3072 high dimension and the second fully connected layer re-projects the 3072 high dimension to he 768 low dimension. Correspondingly, the first and

[1] https://arxiv.org/pdf/2103.00020

second fully connected layer have the weight parameters with size $768 * 3072$ and $3072 * 768$.

After the 12 repeated attention block, there is normalization layer, which has the output $50 * 768$. We extract the first row since the first row is corresponding to the class embedding, not corresponding to any patch. So far the output is the 768-dim vector.

Finally, we have fully connected layer as embedding projection, simply by multiplying 768-dim embedding vector with matrix $768 * 512$, to output 512-dim embedding vector.

The input image is encoded as 512-dim embedding vector.

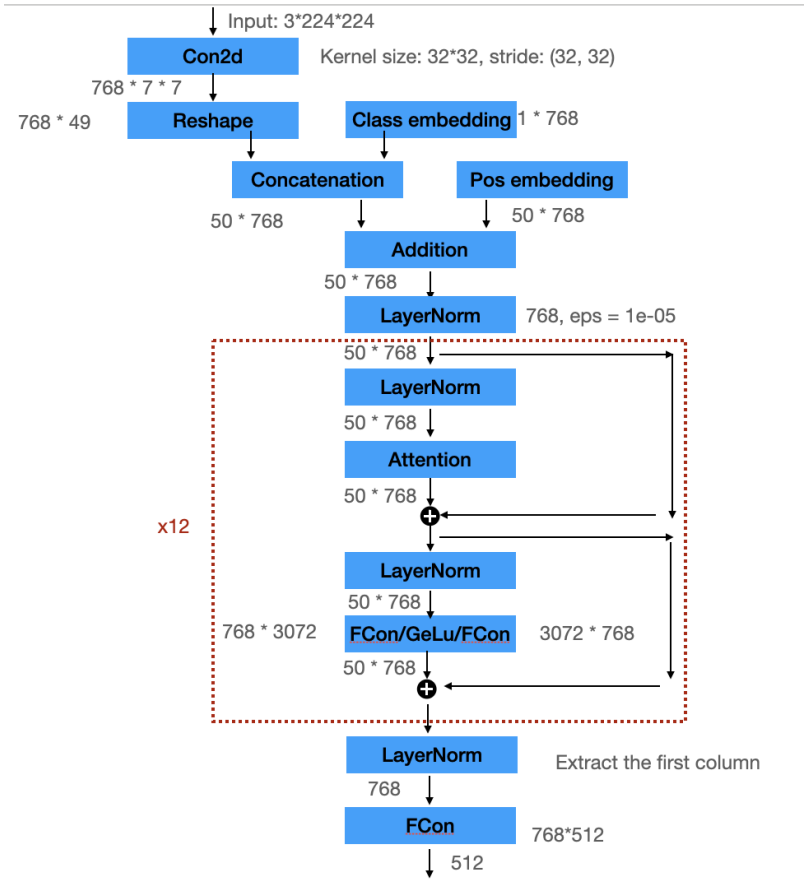

Figure 1: Image Encoder

2 Text Encoder

In the Fig 2, we illustrate how to encode text input to its embedding vector. Assume that text input are 3 phrases : a diagram, a small dog, the large cat. Firstly, based on token id dictionary text input can be converted as

$$\begin{aligned} &\text{a diagram} \rightarrow [49406, 320, 22697, 49407, 0, 0, \dots] \\ &\text{a small dog} \rightarrow [49406, 320, 2442, 1929, 49407, 0, 0, \dots] \\ &\text{the large cat} \rightarrow [49406, 518, 3638, 2368, 49407, 0, 0, \dots] \end{aligned} \tag{1}$$

Where token id 49406 is start token, 49407 is the stop token, 320 represents "a", and 518 is "the" and so on. The maximum length of sentence as input is pre-defined, in this paper, pre-defined as 77, called context length. If the length of the input sentence is less than 77, we pad 0 so that the token vector is same size 77-dim.

The above text input will be converted $3 * 77$ size token matrix with each row representing one phrase. The first step is to convert token id matrix $3 * 77$ to token embedding. In this step, there is token id matrix with size $49408 * 512$ and each column of this token embedding matrix is corresponding to each token. Thus operation is extract each column based token id in token id matrix. After this step, we have $3 * 77 * 512$ tensor representing text input.

To encode positional information, there is the positional embedding matrix with size $77 * 512$, each row 512-dim vector representing one position. Since the maximum length of the input sentence is 77, there are 77 rows in this positional embedding matrix. we add up the $3 * 77 * 512$ token embedding tensor and the $77 * 512$ positional embedding matrix by broadcasting this $77 * 512$ positional embedding matrix in the first dimension of token embedding matrix, then we have output $3 * 77 * 512$ tensor.

Inside the red dash rectangle is attention block. In this specific ViT-B/32 model, there are 12 repeated attention blocks. There are layer normalization layer, multi-head attention layer, fully connected layers, GeLu layers within attention block. For normalization layer input and output have the same size $77 * 3 * 512$. We can easily permute tensor with $3 * 77 * 512$ to tensor with size $77 * 3 * 512$. We omitted the details of attention layer here. The first fully connected layer projects the 512 low dimension to the 2048 high dimension and the second fully connected layer re-projects the 2048 high dimension to he 512 low dimension. Correspondingly, the first and second fully connected layer have the weight parameters with size $512 * 2048$ and $2048 * 512$.

After the 12 repeated attention block and one normalization layer, we have $3 * 77 * 512$ tensor, where 3 is the number of input sentences, 77 is corresponding to context length

(the maximum length of the input sentence) and 512 is final embedding dim. We extract the embedding corresponding the stop token based on the stop token (49407) index. For example, the stop token index for the above three input sentence is $3, 4, 4$, so we extract embedding vector at $3, 4, 4$ position. Thus we have $3 * 512$ matrix where each row is corresponding to each input sentence.

Finally, we have fully connected layer as text projection, simply by multiplying $3 * 512$ embedding matrix with matrix $512 * 512$.

The input 3 sentences are encoded as $3 * 512$ embedding matrix where each row is encode vector for each input sentence.

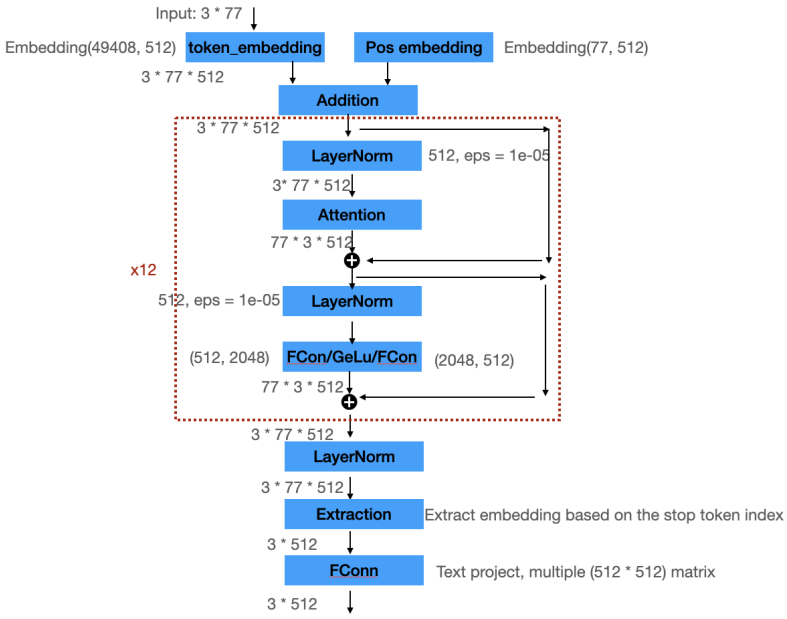

Figure 2: Text Encoder

3 Training Jointly

From the above two sections, we can encode images and text into feature vectors. Assume in one batch we have N (image, text) pairs, where images we denoted as I_i and text T_i, where $i \in \{0, 1, 2, \cdots, N-1\}$. We can convert those image I_i and text T_i to their embedding vectors I_i^e and T_i^e. Based on image and text embedding, we can compute cosine similarity matrix, whose size is $N * N$.

- Each pair image I_i text T_j can be positive pair only and if only $i = j$
- Any other pair image I_i and text T_i is negative pair if $j \neq i$,

Naturally, we can exploit cross entropy loss in this setting as exploited in that paper. For Pseudocode, please refer the original paper.