Attention and Transformer

Junkun Yuan
Zhejiang University
yuanjk@zju.edu.cn

1. Introduction

- **Attention:** catch the important parts. Sec. 2
- **Transformer:** pure attention-based structure. Sec. 3
- **Backbones of Vision Transformer:** the representative backbones of vision transformer. Sec. 4
  - **Vision Transformer (ViT):** apply pure Transformer to sequences of image patches. Sec. 4.1
  - **Swin Transformer (Swin):** shifted window, reduce computation cost. Sec. 4.2
  - **Swin TransformerV2 (SwinV2):** scale up capacity and resolution, move LN to backend, scaled cosine attention, log-spaced continuous position bias. Sec. 4.2
  - **Transformer in Transformer (TNT):** patches as sentences, subpatches as words. Sec. 4.3

2. Attention

In neural networks, **attention** is a technique that mimics cognitive attention. The effect enhances some parts of the input data while diminishing other parts — the thought being that the network should devote more focus to that small but important part of the data. Learning which part of the data is more important than others depends on the context and is trained by gradient descent.[1] In short, **attention** allows the model to pay more "attention" to the important parts of the representations by weighting them.

Self-attention, also called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.[2]

3. Transformer

3.1. What is Transformer?

Transformer is the first transduction model relying entirely on self-attention to draw global dependencies and compute representations without using sequence-aligned RNNs or convolution. (From Sec. 1 and 2 of [6].)

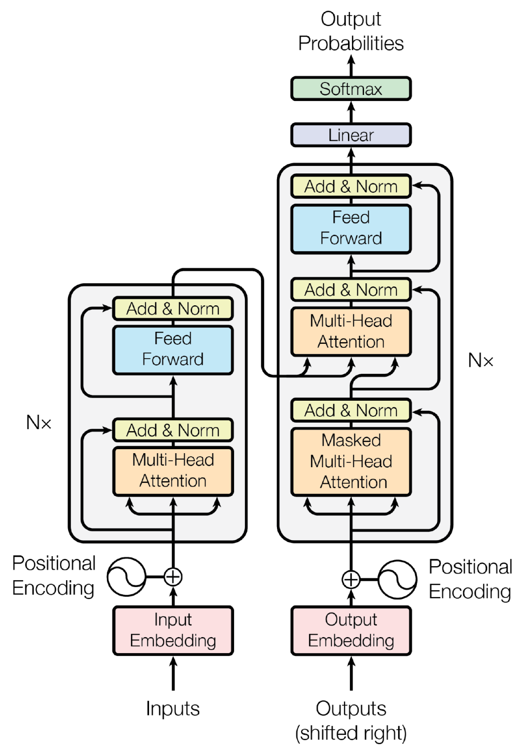

Figure 1. Model architecture of Transformer. (From Fig. 1 of [6].)

3.2. Model Architecture of Transformer

The architecture of Transformer is shown in Fig. 1 and Fig. 2. Good examples to explain Transformer model. [3]

3.2.1 Encoder

The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers: a multi-head self-attention mechanism, and a position-wise fully connected feed-forward network. A residual connection around

[1] From wikipedia: Attention (machine learning).
[2] From wikipedia: self-attention.
[3] Jay Alammar with text in English. Mu Li with video in Chinese.

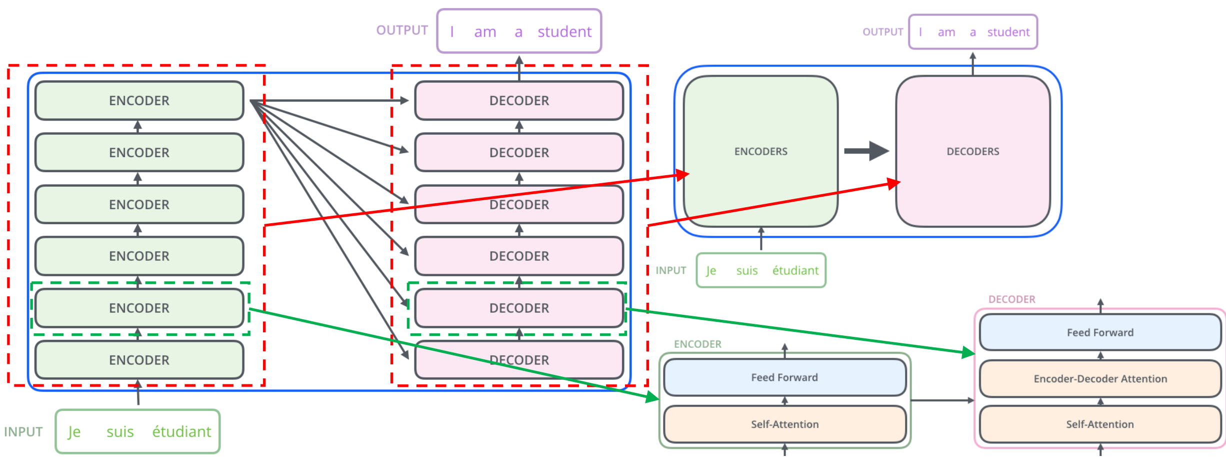

Figure 2. The encoders and decoders of Transformer. (From GitHub: illustrated-transformer.)

each sub-layer is employed, followed by layer normalization. The output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$. Sub-layers and embedding layers produce outputs of dimension $d_{\text{model}} = 512$. (From Sec. 3.1 of [6].)

3.2.2 Decoder

The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, residual connections are employed around each of the sub-layers, followed by layer normalization. The self-attention sub-layer is modified in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i. (From Sec. 3.1 of [6].)

3.2.3 Attention

An attention function maps a query and key-value pairs to an output, which is a weighted sum of the values. The weight assigned to each value is computed by a function of query with the corresponding key. (From Sec. 3.2 of [6].)

Scaled dot-product attention is shown in Fig. 3, its calculation process is shown in Fig. 4. Let queries $Q \in \mathcal{R}^{n\times d_k}$, keys $K \in \mathcal{R}^{m\times d_k}$, and values $V \in \mathcal{R}^{m\times d_v}$, then

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \quad (1)$$

Compared with additive attention [1] that computes compatibility function using a feed-forward network with a single hidden layer, dot product attention is similar in theoretical complexity but is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. For large values of d_k, the dot products grow large in magnitude, pushing the softmax into regions where it has extremely small gradients.

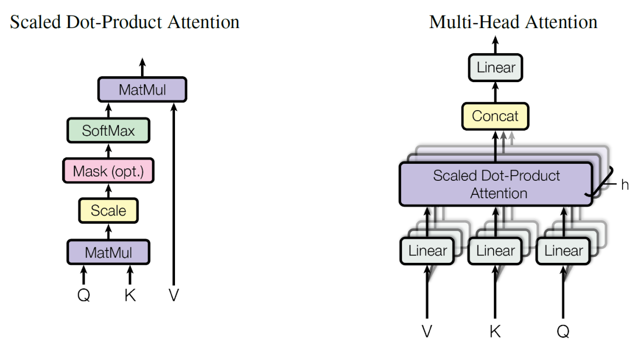

Figure 3. (left) Scaled Dot-Product Attention. (right) Multi-Head Attention. (From Fig. 2 of [6].)

Proof: Let z_i and a_i be the i-th dimension of the input and output of Softmax within K classes, respectively. Thus,

$$\begin{aligned} a_i &= \frac{e^{z_i}}{\sum_{n=1}^{K} e^{z_n}} \\ \text{when } i=j: \frac{\partial a_i}{\partial z_j} &= \frac{e^{z_i}\sum_{n=1}^{K} e^{z_n} - (e^{z_i})^2}{(\sum_{n=1}^{K} e^{z_n})^2} = -a_i^2 + a_i \\ \text{when } i\neq j: \frac{\partial a_i}{\partial z_j} &= \frac{-e^{z_i}e^{z_j}}{(\sum_{n=1}^{K} e^{z_n})^2} = -a_i a_j \end{aligned} \quad (2)$$

When z_i becomes very large under large d_k, a_i tends to 0 or 1, and thus the gradient would become very small. Assume that q and k are independent random variables with mean 0 and variance 1, then $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ has mean 0 and variance d_k, so dividing it by the scaling factor of $\frac{1}{\sqrt{d_k}}$ to normalize it for faster training. (From Sec. 3 of [6].)

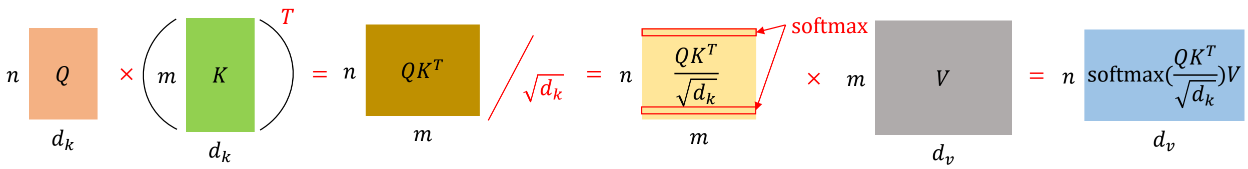

Figure 4. Computation of attention in matrix for Eq. (1). The operations are emphasized in red color.

Multi-head attention allows the model to jointly attend to the information from different representations and subspaces at different positions.

$$\begin{aligned}\text{MultiHead}(Q,K,V) =& \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \\ \text{where head}_\text{i} =& \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned} \tag{3}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$, and $W^O \in \mathbb{R}^{hd_v\times d_{model}}$. Set $h = 8$ parallel attention layers, or heads. For each of these, use $d_k = d_v = d_{model}/h = 64$.

Applications of attention in Transformer. (i) **Encoder-decoder attention layer:** queries come from the previous decoder layer and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. (ii) **Encoder:** keys, values, and queries come from the output of the previous layer in encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder. (iii) **Decoder:** Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. Implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. (From Sec. 3.2.3 of [6].)

3.2.4 Position-wise Feed-Forward Networks

The fully-connected network contains two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{4}$$

The dimensionality of input and output is $d_\text{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.2.5 Embedding & Softmax & Positional Encoding

Use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_model. Use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. Share the same weight matrix between the two embedding layers and the pre-softmax linear transformation. In the embedding layers, multiply the weights by $\sqrt{d_\text{model}}$.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Figure 5. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention. (From Table 1 of [6].)

In order for the model to make use of the order of the sequence, we add positional encodings to the input embeddings at the bottom of the encoder and decoder stacks:

$$\begin{aligned}PE_{(pos,2i)} =& sin(pos/10000^{2i/d_{model}}), \\ PE_{(pos,2i+1)} =& cos(pos/10000^{2i/d_{model}}),\end{aligned} \tag{5}$$

where pos is the position of the word in a sentence and i is the current dimension of the positional encoding. Thus,

$$\begin{aligned}PE_{pos} =& [sin(w_1 \cdot pos), cos(w_1 \cdot pos), ..., \\ & sin(w_{\frac{d_{model}}{2}} \cdot pos), cos(w_{\frac{d_{model}}{2}} \cdot pos)]^T \\ \text{where } w_i =& \frac{1}{10000^{\frac{2i}{d_{model}}}}\end{aligned} \tag{6}$$

The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. It would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

Proof: For any dimension i of a word in position pos:

$$\begin{aligned}PE_{pos+k} =& \begin{bmatrix} sin\,(w_i \cdot (pos + k)) \\ cos\,(w_i \cdot (pos + k)) \end{bmatrix} \\ =& \begin{bmatrix} cos(w_i \cdot k) & sin(w_i \cdot k) \\ -sin(w_i \cdot k) & cos(w_i \cdot k) \end{bmatrix} \begin{bmatrix} sin(w_i \cdot pos) \\ cos(w_i \cdot pos) \end{bmatrix} \\ =& W \cdot PE_{pos}\end{aligned} \tag{7}$$

3.3. Why use Transformer?

See Fig. 5. **Recurrent models** typically factor computation along the symbols of the input and output sequences. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching

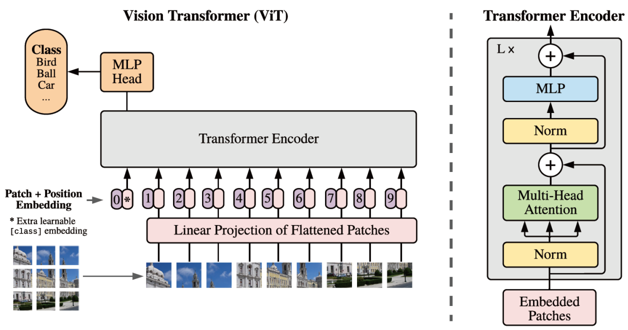

Figure 6. Vision Transformer (ViT) model. (From Fig. 1 of [2].)

across examples. **Convolutional neural networks** compute hidden representations in parallel for all input and output positions, the number of operations required to relate signals from two arbitrary input or output positions grow in the distance between positions, linearly for Conv2S2 and logarithmically for ByteNet. (From Sec. 1 of [6].)

Transformer reduces it to a constant number of operations, albeit at the cost of reduced resolution due to averaging attention-weighted positions. (From Sec. 2 of [6].)

4. Backbones of Vision Transformer

4.1. Vision Transformer (ViT)

Vision Transformer (ViT) [2] directly applies a pure Transformer to sequences of image patches for image classification and other computer vision tasks.

A good example to explain ViT [4].

Fig. 6 shows the model overview. To handle 2D images, we reshape the image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N\times(P^2\times C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. The Transformer uses constant latent vector size D through all of its layers, so we flatten the patches and map to D dimensions with a trainable linear projection. We refer to the projection output as the patch embeddings. We prepend a learnable embedding to the sequence of embedded patches ($\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$), whose state at the output of the Transformer encoder ($\mathbf{z}_L^0$) serves as the image representation $\boldsymbol{y}$. Both during pre-training and fine-tuning, a classification head is attached to $\mathbf{z}_L^0$. The classification head is implemented by a MLP with a hidden layer at pre-training and by a linear layer at fine-tuning. 1D position embeddings are added to the patch embeddings to retain positional information. The MLP contains two layers with GELU.

$$
\begin{aligned}
&\mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D},\\
&\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; ...; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos},\\
&\mathbf{z}'_l = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, l = 1...L,\\
&\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l, l = 1...L,\\
&\boldsymbol{y} = \text{LN}(\mathbf{z}_L^0)
\end{aligned} \tag{8}
$$

[4] Yi Zhu with video in Chinese.

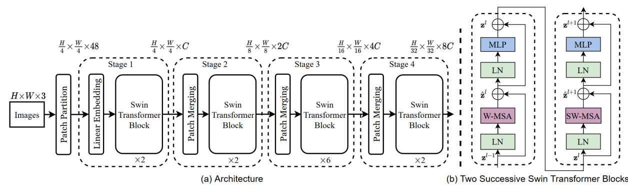

Figure 7. Swin Transformer architecture. (From Fig. 1 of [5].)

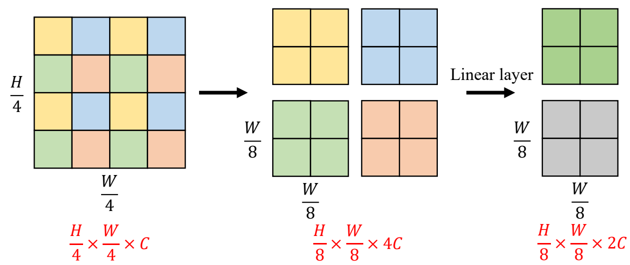

Figure 8. Patch merging of Swin Transformer in stage 2.

4.2. Swin Transformer (Swin)

Swin Transformer [5] is proposed to serve as a general-purpose Transformer-based backbone for computer vision. A good example to explain Swin Transformer. [5]

The architecture of Swin Transformer is shown in Fig. 7. Patch size is 4×4 and feature dimension of each patch is $4\times 4\times 3 = 48$. A linear embedding layer projects it to arbitrary dimension C. Let $\hat{\mathbf{z}}^l$ and $\mathbf{z}^l$ denote the output features of the (S)W-MSA ((shifted) window multi-head self-attention) module and the MLP module for block l, respectively.

$$
\begin{aligned}
\hat{\mathbf{z}}^l &= \text{W-MSA}(\text{LN}(\mathbf{z}^{l-1})) + \mathbf{z}^{l-1}\\
\mathbf{z}^l &= \text{MLP}(\text{LN}(\hat{\mathbf{z}}^l)) + \hat{\mathbf{z}}^l\\
\hat{\mathbf{z}}^{l+1} &= \text{SW-MSA}(\text{LN}(\mathbf{z}^l)) + \mathbf{z}^l\\
\mathbf{z}^{l+1} &= \text{MLP}(\text{LN}(\hat{\mathbf{z}}^{l+1})) + \hat{\mathbf{z}}^{l+1}
\end{aligned} \tag{9}
$$

The first **patch merging** layer concatenates the features of each group of 2×2 neighboring patches, and applies a linear layer to reduce the tokens from $4C$ to $2C$. See Fig. 8.

Computational complexity is (see Fig. 9):

$$
\begin{aligned}
\Omega(\text{MSA}) &= 4hwC^2 + 2(hw)^2C\\
\Omega(\text{W-MSA}) &= 4hwC^2 + 2M^2hwC
\end{aligned} \tag{10}
$$

[5] Yi Zhu with video in Chinese.

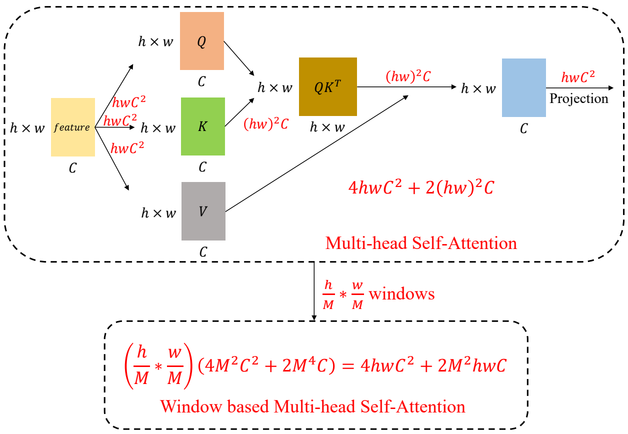

Figure 9. Computation complexity of multi-head attention (above) and window-based multi-head self-attention (below). h: height, w: width, C: channel, each window contains $M \times M$ patches.

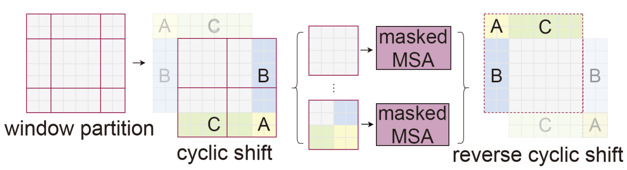

Figure 10. Efficient batch computation approach for self-attention in shifted window partitioning. (From Fig. 4 of [5]).

Efficient computation of **shifted window** see Fig. 10.

Swin Transformer V2 [4] aims to scale up model capacity and resolution. To make the training of large models more stable, it moves LN layer from the beginning of each residual unit to the backend to make activation value milder, and uses scaled cosine attention instead of dot product to make the computation irrelevant to amplitudes of block input. To enable large window size, it uses a log-spaced continuous position bias, which generates bias values for arbitrary coordinate ranges by applying a meta network on the log-spaced coordinate inputs. To resolve the memory issue, it incorporate several techniques including zero-optimizer, activation check pointing, and a novel implementation of sequential self-attention computation. (From Sec. 1 of [4])

4.3. Transformer in Transformer (TNT)

TNT [3] divides the input images into several patches as "visual sentences" and further divide them into sub-patches as "visual words". The Transformer blocks are applied to the inner and outer parts. A good example to explain it. [6]

Given a 2D image, split it into n patches $\mathcal{X} = [X^1, X^2, ..., X^n] \in \mathbb{R}^{n \times p \times p \times 3}$, where $p \times p$ is patch resolution. Each patch is further divided into m sub-patches:

$$X^i \rightarrow [x^{i,1}, x^{i,2}, ..., x^{i,m}], \tag{11}$$

where $x^{i,j} \in \mathbb{R}^{s \times s \times 3}$ is the j-th visual word of the i-th visual sentence. Embed the words with linear projection:

$$Y^i = [y^{i,1}, y^{i,2}, ..., y^{i,m}], y^{i,j} = FC(Vec(x^{i,j})), \tag{12}$$

where $y^{i,j} \in \mathbb{R}^c$ is the j-th word embedding, c is the dimension of word embedding, and $Vec(\cdot)$ is the vecterization operation. Use a transformer block for visual words:

$$\begin{aligned} Y_l'^i &= Y_{l-1}^i + MSA(LN(Y_{l-1}^i)), \\ Y_l^i &= Y_l'^i + MLP(LN(Y_l'^i)). \end{aligned} \tag{13}$$

Add the words to sentences:

$$Z_{l-1}^i = Z_{l-1}^i + FC(Vec(Y_l^i)). \tag{14}$$

Use the standard transformer block for sentences:

$$\begin{aligned} Z_l' &= Z_{l-1} + MSA(LN(Z_{l-1})), \\ Z_l &= Z_l' + MLP(LN(Z_l')). \end{aligned} \tag{15}$$

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 4

[3] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34, 2021. 5

[4] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021. 5

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 4, 5

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 2, 3, 4

[6] Han Kai (the author) with video in Chinese.