ViT for 1D spectra recognition

Lorenzo Pasquetto

November 2024

Abstract

The use of convolutional layers in deep learning approach to spectra recognition has become a standard for its efficiency. Indeed the CNN has revolutionized vision algorithms for application from 1D spectra to 3D video data. In the following, we have constructed a 1D version of the well known ViT from [1], one of the first sequential approach to image recognition.

1 Introduction

The core of ViT is the concept of transformer and the use of attention mechanism introduced by Ashish Vaswani et al. in [2]. The first application of attention was in the context of language model and machine translation. Before transformers, language translation models processed input sequentially, handling one word or token at a time to translate a phrase from one language to another. This means that the neural network (NN) must have a memory of the previous inputs. This is done using recurrent neural networks (different from feed forward networks) which process data across multiple time steps. An example of well established RNN ConvS2S. The general architecture can be synthetized in two main blocks, an Encoder and a Decoder. The first one consists of RNN-based layers like Long-Short-Term-Memory. It takes the input token-by-token sequentially and update an hidden state h_t which represent the information of the sentence (the context vector). The second block is also RNN-based. It takes the context vector and generates the target sequence token-by-token. It is clear that the performance of the model strongly depends on the amount of information encoded in the context vector. Another problem which arise from RNN-based networks is related to the sequential nature of encoding-decoding process. This leads to low training and inference time due low parallelization.

2 Multi-Head Attention

Everything changes with the use of Multi-Head attention. An attention can be described as mapping a query (the current token we are focusing on) and a set of key-value pairs (tokens in the sequence we are comparing against) to an output. Basically we have three matrices in which the corresponding the rows are the embeddings of word in the phrase. Suppose we have three words phrase: "*the child eats*". If we want to know the attention score of child in the phrase we multiply the embedding of child (suppose for simplicity $[0, 1, 0, 1]$) with the matrix of the embeddings of the words. We scale this vector with the square root of the keys dimension d_k (in this case 2), apply a softmax and multiply for the value matrix which is the same of keys matrix:

$$Attention(Q_{cat}, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \cdot V \quad \text{where: } Q \in \mathbb{R}^{1\times4}, K \in \mathbb{R}^{3\times4}, V \in \mathbb{R}^{3\times4}$$

The attention is then a 1×4 vector. We can repeat the process for each word in the phrase obtaining a 3×4 matrix of the attention scores. This type of attention is called scaled dot-product attention. In [2], Ashish Vaswani et al. proposed a different attention based on the same principle. In this attention the keys, queries and values are linearly projected h times into different spaces (d_k, d_k and d respectively). This is done by a simple matrix multiplication with matrices W^Q, W^K and W^V, where the entries corresponds to learnable parameters.

$$\begin{aligned} Q^{proj} &= Q \cdot W^Q, \quad \textbf{where } W^Q \in \mathbb{R}^{d_{model}\times d_{head}} \\ K^{proj} &= K \cdot W^K, \quad \textbf{where } W^K \in \mathbb{R}^{d_{model}\times d_{head}} \\ V^{proj} &= V \cdot W^V, \quad \textbf{where } W^V \in \mathbb{R}^{d_{model}\times d_v} \end{aligned}$$

Each of the projected version of the queries, keys and values passes through an attention mechanism producing h different outputs. We call these outputs heads. The heads are then concatenated, reassemble the initial dimensionality d_{model}. Each attention head learns its own set of projections (W^Q, W^K and W^V), enabling it to focus on different relationship in the input sequence.

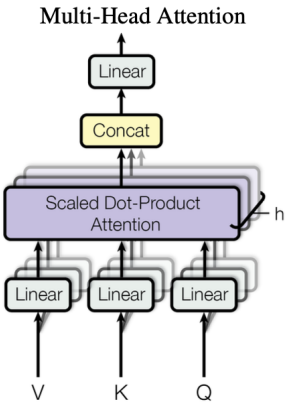

Figure 1: Multi-Head Attention scheme, image from [2]

.

Exploiting the dot-product, we have successfully created a block which can evaluate the relationship between two words no matter the distance (non locality).

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h) \cdot W^O$$
$$\text{where : } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

The multi-head attention score is the added to the input embeddings and then normalized. The normalized output is passed to a feed forward network and then normalized and add to a residual connection (see figure (2)).

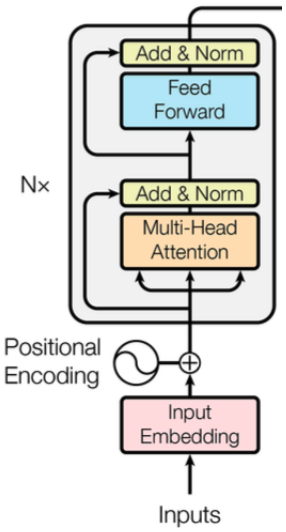

Figure 2: Scheme of the original transformer, image from [2]. Note that, in language models we use also use a decoder to obtain the translation.

3 Transformer-based model in 1D case

Since the good performance of the general attention block introduced in our convolutional neural network (see NonLocalBlock class), we decide to implement a new model for regression task. We get as input an array of 4500 elements. This array constitutes a powder X-ray diffraction spectra and from it we want to determine the crystal parameters. The new model must be convolution-free. The point is to define what is a token for the transformer. We decide to follow the idea of Alexey Dosovitskiy et al., in [1], and create patches from the initial spectra. These patches are then projected to higher space. As usually done in LLM like BERT, we define an additional token called "`cls`" (which stands for *classification*) which store the information of the spectra. This `cls` token is initialized as a learnable parameter, allowing the model to represent the spectra (or the important features) in this token. The information in `cls` is store dynamically through the self-attention layer in the transformer. This avoid to use very deep models, i.e. the vanishing gradient problem is overcome.

The self-attention mechanism calculates an attention score between a query token and all the other tokens in the sequence. This parallel computation lacks inherently of positional sense (any token is spatially "equivalent"). To reconstruct this information we can modify the patch embeddings whit a positional encoder. We need to create an encoding for each patch and that must be unique. Usually this is done by meant of sine and cosine functions applied at even and odd position respectively. In figure, we show an heat map of the encoding for 512 positions (patches) each one of dimension 10 (embedding dimension).

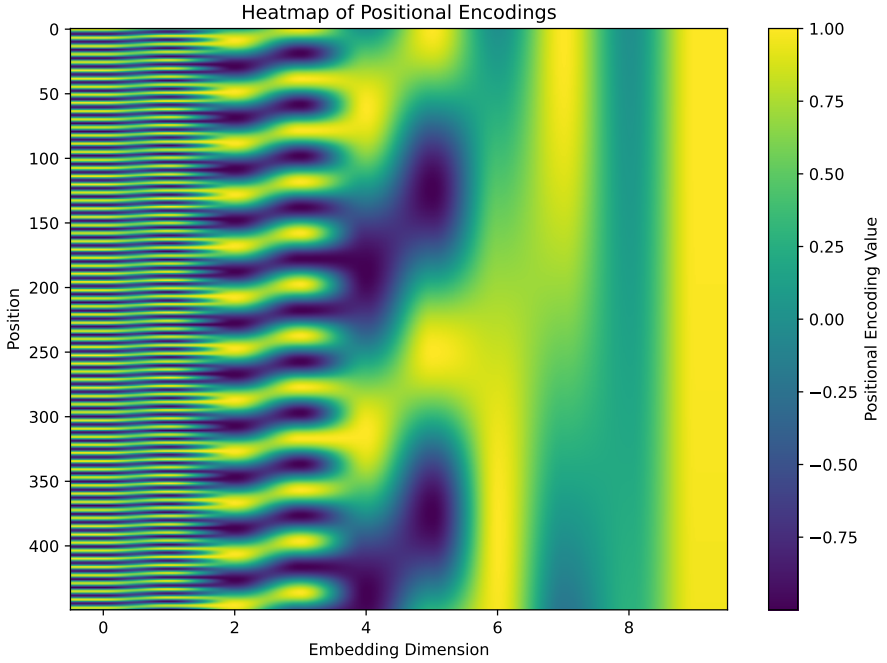

Figure 3: Heat map of the encoding for 512 positions each one of dimension 10. In simple terms, at the patch number 300 we add the row 300 of the heat map. This is done for all the tokens (i.e. patches in our model). Such method ensures a unique encoding for the positions.

Actually in our model, we use a **learnable positional encoding**. New learnable parameters are initialized with shape (1, `num_patches` + 1, `embedding_dim`) and added to the the different patches. The model now learns the positional representations of the tokens as parameters during the train. In simple term a new matrix L is initialized such that:

$$L \in \mathbb{R}^{\texttt{num\_patch} \times \texttt{emb\_dim}} \longrightarrow (\texttt{num\_patch} \times \texttt{emb\_dim}) \ \texttt{new learnable parameters}$$

It contains the parameters that can be modified during training and added to the embeddings of the tokens. As before, you add the i-th row of L to the i-th token.
Now the tokens are ready to pass through transformer layers. The output represents a contextualized embedding for each patch. Since we decided to add the `cls` tokens we exploit it to gather global information of the spectra. This means that, we pass the `cls` directly the the multi-layer perceptron

(MLP). The MLP is a simple sequential block of two dense layers, one of 64 units and one with six units (our output). The `cls` token is normalized before MLP.

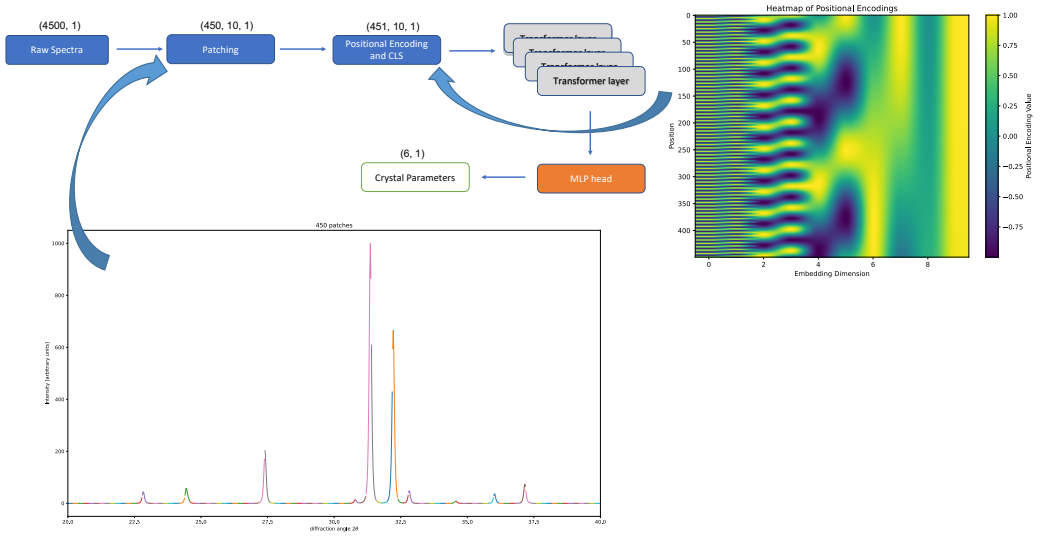

Figure 4: Example of sequences for 450 patches using four transformer encoder layers.

Table 1: Transformer-based model

System	MAPE	MAPE_{CNN}	R2
Triclinic	8.72%	6.70%	0.31

References

[1] 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. *CoRR*, abs/2010.11929, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.