Summary of ViViT: A Video Vision Transformer

Ashwin Ramachandran, Akshat Kumar

October 20, 2023

1 Intorduction

The paper presents a pure transformer based model for video classification. They extract spatiotemporal tokens from the input video, and then encode them using a transformer. They propose different ways to factorise the spatial- and temporal-dimensions of the input. They show that their method outperforms prior methods based on deep 3D convolutional networks.

2 Video Vision Transformers

2.1 Overview of Vision Transformers (ViT)

The vision transformer processes 2D images by dividing them into N non-overlapping image patches. Each image patch $x_i \in R^{hxw}$ is projected using a 2D convolution to a single dimension token $z_i \in R^d$. The N tokens are prepended by a learned classification token z_{cls} and added a learned position embedding $p \in R^{Nxd}$. The sequence is then input to a transformer encoder. The representation of z_{cls} at the final layer of the encoder serves as the final representation used by the classification layer. The position embedding is used to aid self-attention operations in the transformer since they are permutation invariant. Finally, a linear classifier is used to classify the encoded input based on $z_{cls} \in R^d$.

2.2 Embedding Video Clips

Two simple methods are presented for mapping a video $V \in R^{THWC}$ to a sequence of tokens $z \in R^{ntnhnwd}$. The first method uses the transformer to fuse temporal information from different frames whereas the second method fuses the spatio-temporal information during tokenisation.

2.2.1 Uniform frame sampling

First, n_t frames are uniformly sampled from the input video clip. Next, each 2D frame is divided into $n_h x n_w$ non-overlapping image patches, which are em-

bedded in the same way as in ViT to give a total of $n_t x n_h x n_w$ tokens. This is then forwarded through the transformer encoder.

2.2.2 Tubelet sampling

This method extracts non-overlapping, spatio-temporal "tubes" from the input volume, and then linearly projects this to R^d. If we assign a "tubelet" a dimension of thw, then in total $n_t x n_h x n_w$ tokens where $n_t = \lfloor \frac{T}{t} \rfloor$, $n_h = \lfloor \frac{H}{h} \rfloor$ and $n_w = \lfloor \frac{W}{w} \rfloor$, are extracted.

2.3 Transformer Models for Video

2.3.1 Spatio-temporal attention

In this model, all spatio-temporal tokens extracted from the video are forwarded through the transformer encoder. The pair-wise interactions in transformer between tokens, helps model long-range interactions across the video. The disadvantage with this model is that, it has quadratic complexity with respect to the number of tokens, and hence with the increase in number of frames in the video.

2.3.2 Factorised encoder

This model employs two distinct transformer encoders in its architecture. The first encoder, known as the spatial encoder, specifically focuses on capturing interactions between tokens that are extracted from the same temporal index. The representation for each temporal index, represented as $h_i \in R^d$ is the representation of z_{cls}. These frame-level representations, h_i, are then combined into a matrix $H \in R^{n_t d}$ and subsequently passed through a temporal encoder, which is designed to model interactions between tokens from different temporal indices. The output token of this encoder is then employed for the final classification task.

Interpretation: The spatial encoder is typically utilized in image classification tasks. It follows a pattern observed in CNN architectures where per-frame features are initially extracted and then aggregated into a final representation before classification. Despite having more transformer layers compared to Model 1, and thus more parameters, this model is computationally more efficient in terms of floating-point operations (FLOPs). This efficiency is achieved due to the two separate transformer blocks, which have a complexity of $O((n_h x n_w)^2 + n_t^2)$, as opposed to the complexity of $O((n_t n_h n_w)^2)$ seen in Model 1.

2.3.3 Factorised self-attention

This model, in contrast, contains the same number of transformer layers as Model 1. However, instead of computing multi-headed self-attention across all pairs of tokens, at layer l, we factorise the operation to first only compute self-attention spatially (among all tokens extracted from the same temporal index),

and then temporally (among all tokens extracted from the same spatial index) as shown in Fig. 5.

Each self-attention block in the transformer thus models spatio-temporal interactions, but does so more efficiently than Model 1 by factorising the operation over two smaller sets of elements, thus achieving the same computational complexity as Model 2.

This operation can be performed efficiently by reshaping the tokens z from $R^{1\times nt\cdot nh\cdot nw\cdot d}$ to $R^{nt\cdot nh\cdot nw\cdot d}$ (denoted by z_s) to compute spatial self-attention. Similarly, the input to temporal self-attention, z_t, is reshaped to $R^{nh\cdot nw\cdot nt\cdot d}$.

Here we assume the leading dimension is the "batch dimension". Our factorised self-attention is defined as:

$$y_s^l = MSA(LN(z_s^l)) + z_s^l \tag{1}$$

$$y_t^l = MSA(LN(y_s^l)) + y_s^l \tag{2}$$

$$z^{l+1} = MLP(LN(y_t^l)) + y_t^l \tag{3}$$

2.3.4 Factorised Dot-Product Attention

In this section, we introduce a model that maintains the same computational complexity as Models 2 and 3, while retaining the parameter count of the unfactorised Model 1. Similar to Model 3, we factorise the spatial and temporal dimensions, but we introduce factorisation within the multi-head dot-product attention operation.

In this model, we calculate attention weights for each token individually, considering the spatial and temporal dimensions separately using different attention heads. The attention operation for each head is defined as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{4}$$

In self-attention, the queries $Q = XW_q$, keys $K = XW_k$, and values $V = XW_v$ are linear projections of the input X with $X, Q, K, V \in R^{N\times d}$. Note that in the unfactorised case (Model 1), the spatial and temporal dimensions are merged as $N = nt \cdot nh \cdot nw$.

The main concept here is to modify the keys and values for each query in such a way that they attend only to tokens from the same spatial and temporal index. We construct $K_s, V_s \in R^{nh\cdot nw\times d}$ and $K_t, V_t \in R^{nt\times d}$, representing the keys and values corresponding to these dimensions. For half of the attention heads, we attend to tokens from the spatial dimension by computing $Y_s = Attention(Q, K_s, V_s)$, and for the remaining heads, we attend to the temporal dimension by computing $Y_t = Attention(Q, K_t, V_t)$. Since we change the attention neighborhood for each query, the attention operation maintains the same dimensions as the unfactorised case, i.e., $Y_s, Y_t \in R^{N\times d}$. We then combine the outputs of multiple heads by concatenating them and applying a linear projection, denoted as $Y = Concat(Y_s, Y_t)W_o$.