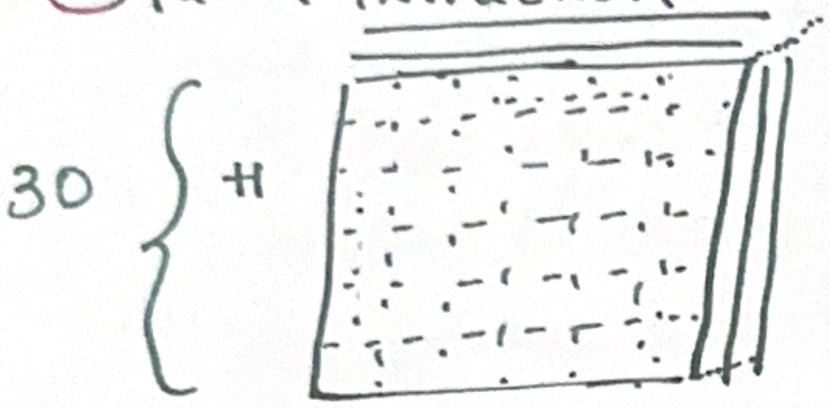
Visiontransformer: AN IMAGE is WORTH 16×16 words:

① Input Image $X \in \mathbb{R}^{H \times W \times C} = \mathbb{R}^{480 \times 480 \times 3}$

Height, width, channels

② Patch Extraction



X = Input image $\in \mathbb{R}^{480 \times 480 \times 3}$

The image is divided into patches

$(H/P)(W/P)$ P is the Size of the patches

P = 16×16 - Patch Size

In our Example we get

$\left(\frac{480}{16}\right) * \left(\frac{480}{16}\right) = 30 \times 30 = 900$ patches.

Now we put these 900 patches in a Row

and Each patch has (Patch size_x)(Patch_Size_y)(channels)

[16 × 16 × 3 = 768]

This Gives us patch embedalings.

$Patchembd \in \mathbb{R}^{900 * 768}$

Image X $\xrightarrow{\text{patches}}$ patch embedding. flatten

$\mathbb{R}^{480 \times 480 \times 3} \longrightarrow \mathbb{R}^{900 \times 768}$

$R^{H \times W \times 3} \longrightarrow \mathbb{R}^{\#ofpatches \times (p \times p \times c)}$

p = patch size

C = #of channels

N = # of patches when inaage is divided.

③ Linear projecton of flatten Spaces.

A learnable linear projection is applied to the flatten palch embeddings.

$patchembedalinge \boxed{\mathbb{R}^{N \times (p^2 \times c)}} * W_f \in \mathbb{R}^{(p^2 \times c) \times D}$ "D" can be 512 or any dim

Shape After Operation.

$\mathbb{R}^{N \times D}$ where N = #of atches

D = 512 or Any dimension.

④ Positional Encoding

we have divided the Image into patches and now we are Adding positional encoding to these flatten patches that are projected to D dimensions.

why Bother with positional encoding?.

we have flatten our image and stacked all the patches in Row Now there is no information preserved about the position of the patch no spatial information is Retained when the patches are flatten we just multiply the $(h' * w' \times 3)$ here $h' = w' =$ patch size = P and stack them

Adding positional encoding to the flatten patches to maintain the position information among the patches.

learnable

The dimension of the position Encoding is equal to the flatten projected patch size

$\in \mathbb{R}^{N \times D}$

flatten projected patch $\oplus$ positional Encoding $\rightarrow$ positionencoded embd

$\mathbb{R}^{N \times D}$ $\quad$ $\oplus$ $\quad$ $\mathbb{R}^{N \times D}$ $\quad$ $\mathbb{R}^{N \times D}$

⑤ Adding CLS_token

Since we are trying to predict the "image class" with "transformer." ViT we want to add An Classification class to the flatten outputs Since this will help us make the prediction about the class we use this for our loss and optimizer will update the parameters ~~st~~ for this task.

Stack

pos-encoded-embedding $\oplus$ CLS-token $\rightarrow$ Input to transformer

$\in \mathbb{R}^{N \times D}$ $\quad$ $\in \mathbb{R}^{1 \times D}$ $\quad$ $\in \mathbb{R}^{(N+1) \times D}$ $\quad$ +1 = CLS-token

⑥ Transformer Encoder Layer:

Input to the transformer Encoder layer is [Flatten patches · projected to $\mathbb{R}^{N\times D}$ D' dimension ⊕ positional encoding] Stack Cls_token for class prediction. $\mathbb{R}^{(N+1)\times D}$

→ Input to transformer $\mathbb{R}^{(N+1)\times D}$

→ Linear Projection (Query, key, values) (Q, K, V) $(W_Q \; W_K \; W_V)$ — Q, K, V: $\mathbb{R}^{(n+1)\times d}$; W_Q, W_K, W_V: $\mathbb{R}^{D\times D}$

→ multi head attention Concatenate $\left[\left(\text{Softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right) * V\right)\right]$

→ first (QK^T) → Query into Keys Heads

→ Second $\left(\frac{QK^T}{\sqrt{d_K}}\right)$ → $\sqrt{d_K}$ for numerical Stability

→ Third Softmax $\left(\frac{QK^T}{\sqrt{d_K}}\right)$ Attention Scores

→ fourth (Softmax x V) Projections.

→ fifth Concatenate (Softmax x V) Concatenate $\mathbb{R}^{(n+1)\times d_K \times n}$

$\left(\frac{QK^T}{\sqrt{d_K}}\right) * V$: $\mathbb{R}^{(n+1)\times(n+1)} \times \mathbb{R}^{(n+1)\times d}$ → $\mathbb{R}^{(n+1)\times(n+1)} \times \mathbb{R}^{(n+1)\times d}$

$d_K = \frac{D}{n}$

n = no of heads in MHA

⑦ Residual Connections & LayerNorm

The Output of the Attention/MHA is added is Added Back to the input and this Residual Connection is then passed through a layer normalization

layer norm [Input to transformer + Output of multi head attention]

Concatenated heads

$\in \mathbb{R}^{(n+1)\times D}$ ⊕ $\in \mathbb{R}^{(n+1)\times d_K \times n} = \mathbb{R}^{(n+1)\times D}$ (pointwise added matrix)

pointwise

↓ layer Norm

$\mathbb{R}^{(n+1)\times d}$

⑧ Feed forward network

1) first linear layer (Expand) $[\text{layer Norm}] \times [W\_\text{first linear projection}] = \mathbb{R}^{(n+1)\times D} \times \mathbb{R}^{D\times \text{Expand}} = \mathbb{R}^{(n+1)\times \text{Exp}}$

Relu Activation (Activation function) $\text{Relu}[\text{first}] = \max(0, \text{value}) = \mathbb{R}^{(n+1)\times \text{Exp}}$

second linear layer (Contract Back) $[\text{Relu}] \times [\omega\_\text{second\_linear.projection}]$

$$\mathbb{R}^{(n+1)\times \text{Expand}} \times \mathbb{R}^{\text{Expand}\times D} \rightarrow \mathbb{R}^{(n+1)\times D}$$

Residual Connection & layer Normalization $\in \mathbb{R}^{(n+1)\times D}$

Apply layer Norm (fNN_second linear_layer $\in \mathbb{R}^{(n+1)\times D}$ ⊕ Normalized outputs from layer Norm Block Before fNN)

The output of fNN is added to the Normalized output from the previous layer pointwise

After the point wise addition we Apply layernorm to the ~~out~~ pointwise Sum.

The dimension After the layernorm to the pointwise Sum Remains the Same $\mathbb{R}^{(n+1)\times D}$

This is the output of the Encoder layer. Transformer $\mathbb{R}^{(n+1)\times D}$

⑨ Classification:

Now from the output of transformer_encoder_output

$$\in \mathbb{R}^{(n+1)\times D}$$

we extract the cls-token vector

Since this CLS_token vector was a traincable parameter during training we take this vector

$$\mathbb{R}^{1\times D} = \text{CLS\_token (vector)}$$

Then we pass this through a linear_layer_mlp that is of dimension of $\mathbb{R}^{D \times \text{numberof classes.}}$

$$\text{cls\_token} \times \text{weight\_for\_last\_mlp} =$$

$$\mathbb{R}^{1\times D} \times \mathbb{R}^{D\times \text{no of classes}} = \mathbb{R}^{1\times \#\text{of classes}}$$