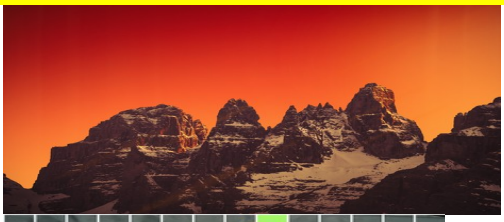
Process

Input Image of shape (224, 224, 3)



Patch Size for image set to 16 x 16

Number of patches per image set to 196 (224/16=14x14)

Text input: " a red ball on the grass"

Tokenizer tokenizes: "a", "red", "ball", "on", "the", "grass", "."

Append [SOS] and [EOS] token, so the total text tokens becomes t + 2 special tokens.

Number of attention heads set to 12 and the subspace size for each head is 64-dim

Vision Encoder

Flatten patch 16x16x3 = 768-dim vector, its just pure reshaping operation without any learnable parameters

Project to D = 768 through linear project, this is a matrix multiplication of (1, 768) vector with (768, 768) dimensional matrix

Adds positional embedding P belongs to Real Number (196 x768) (rows x column)

there are 196 position vector for 196 image patches each with the dimension of 768

Adds to patch embedding E(patch) = PatchEmb + P (belongs to [Real Number (196x768)

P means just patch embeddings, each of the position embeddings are added to the respective embedding of the patch.

Adds [CLS] token which acts as a summarizer of that token space. And it is learnable vector of dimension [Real Number (768)]

its like a summary vector which increases the row size from 196 to 197.

Transformer

Layer Normalization to the inputs using X(norm) = LayerNorm(X(vis) belongs to [Real Number (197x768)].

layer normalization normalizes/stabilize each patch's token independently across 768 dimension.

Project to Q, K, V, where Q = X(norm)W(q), K = X(norm)W(k) and so on.

Each W belongs to [RealNumber(768 x 768)] and Q, K, V belongs to [RealNumber(197x768)

Reshape (split into heads) Q(h), K(h), V(h) and they belongs to [RealNumber(197x64)] for each of 12 heads.

(197x768) matrix converted into 197, 12, 64) and transposed to (197 x 64) for each of 12 heads.

$$\text{Attention}^h = \text{softmax}\left(\frac{Q^h(K^h)^\top}{\sqrt{64}}\right) \in \mathbb{R}^{197\times197}$$

$$\text{Output}^h = \text{Attention}^h V^h \in \mathbb{R}^{197\times64}$$

Computer attention per head, and the Output(h)

Concatenates MHA = Concat(output1, output2….outputR)
belongs to [RealNumber(197x768)]

Projects with W(o) like this: MHA(out) = MHA x W(o)
belongs to [RealNumber(197 x768)]

Wo is a 768x768 dimensional vector which allow combining the information from all the heads intelligently instead of just mere concatenation.

Vector	Matrix
vector is a single token like a row or column containing 768 number	Matrix is a grid containing multiple rows and columns, where rows means number of items and columns means features per items, like 196 x 768. Vectors are stacked as rows and columns in matrix
its a single egg	its a tray of 100 eggs
For A·B to work, the columns of A must equal rows of B.	like this: (1, 768) . (768 x 768) but can't happen same when, (1, 768). (1, 768)

For each 768 values, the mean and std is calculated and later, the substracted by mean and divided by std

Each token is multiplied by the matrix of 768 x 768, and this is a vector x matrix multiplication.

- $Q^h \in \mathbb{R}^{197\times64}$
- $K^h \in \mathbb{R}^{197\times64}$
- $(K^h)^\top \in \mathbb{R}^{64\times197}$
- So $Q^h(K^h)^\top \in \mathbb{R}^{197\times197}$

K is transposed due to which the dimension changes to (197 x 197) then downscales using square root and softmax turn this vector into a probability distribution.

Weight Matrix

a special kind of matrix that transforms one vector to another, they are trainable and their value are updated during back propagation.

Its shape is D(in) x D(out), what it takes and what is produces.

its a recipe to create a omelette.
If You have a 768-d input vector → Din=768.You want a 768-d Query vector → Dout=768. So weight matrix = 768×768

$$\text{Output}^h = \text{Attention}^h \cdot V^h \in \mathbb{R}^{197\times 64}$$

This gives the new matrix of dimension (197x64) again which is the output of that atttention head.