Vision Transformer

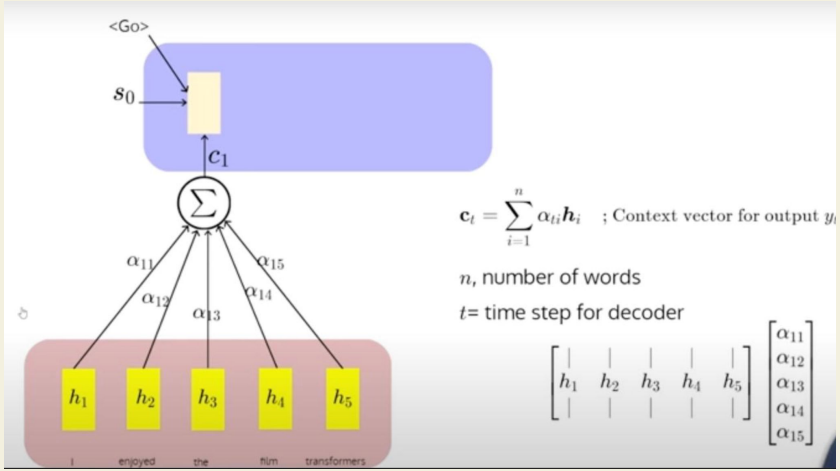
By Jayin Khanna

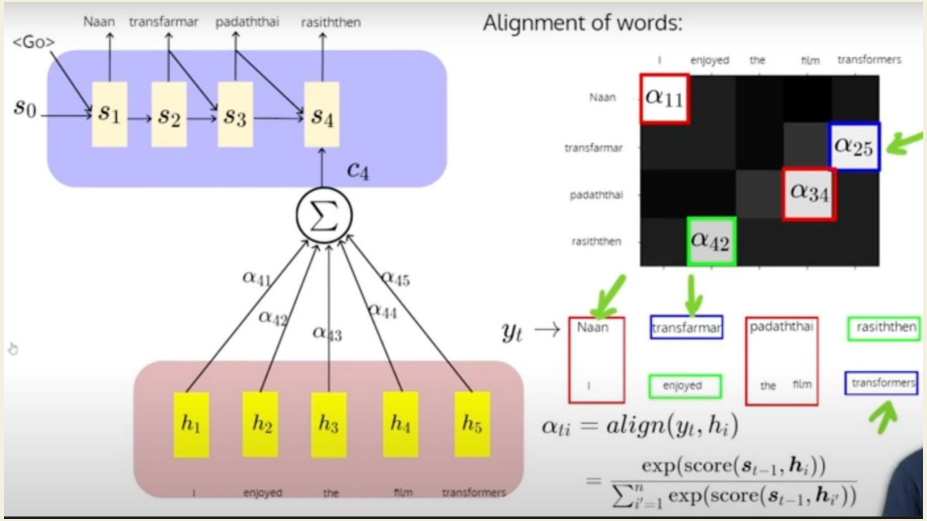
1. Defining the Problem
2. **Making Observations**
3. Forming a Hypothesis
4. Experiment Results
5. Drawing a Conclusion

Attention and transformers

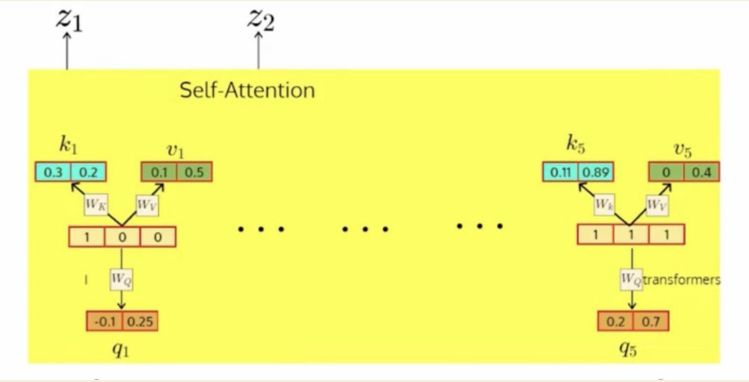
Why not RNNs:

1. We want contextual representation ie. we want an improved representation based on context and
2. We don't want sequential processing; instead we want parallelizability in the architecture

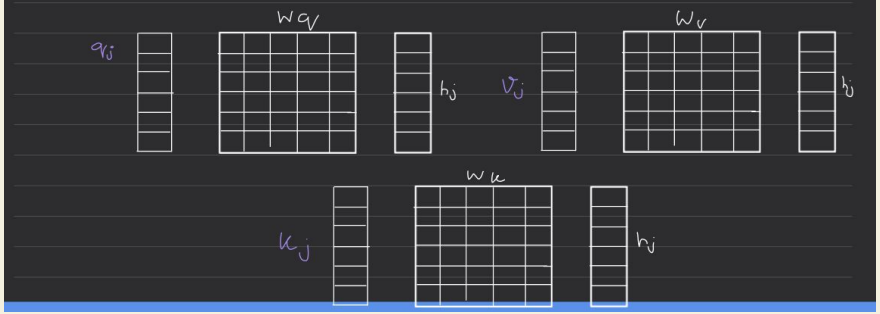




Where H_i: the word embedding of the ith token in the sentence. And alpha is the contribution of each of the other token embeddings on the current token embedding



*) Tranformation Matrices:



*) Attention Mechanism:

Given: $X = \{x_1, x_2, \cdots, x_T\}$; $x_i \in \mathbb{R}^d$

→ first compute three sets of representations
i) key
ii) query & iii) value

let input be $X \in \mathbb{R}^{T \times d}$

projections q, k, v:

$$q = \underset{d \times d_q}{X W_q} \quad , \quad k = \underset{d \times d_k}{X W_k} \quad , \quad V = \underset{d \times d_v}{X W_v}$$

typically $d_q = d_k = d_v$

Each row of the above matrices corresponds to the representations of every single token.

Attention:

given a query vector $q \in \mathbb{R}^{d_v}$, (a row of q)

a set of all the key & value vectors
$k_i \in \mathbb{R}^{d_v}$, $v_i \in \mathbb{R}^{d_v}$ (rows of matrices k & V)

1.) find the similarities b/w the given query & all the keys

$S = q^T k^T$ where given query $q \in \mathbb{R}^{1 \times d_v}$
$k \in \mathbb{R}^{T \times d_v}$
hence, $S \in \mathbb{R}^{1 \times T}$
S_i : similarity b/w q & k_i

2.) Scale the dot product to normalize

$$\hat{s} = \frac{q^T k^T}{\sqrt{d_v}}$$

3.) normalize values of $\hat{s}$ to have values b/w 0 & 1 & sum to 1

Softmax $(\hat{s}) : \mathbb{R}^T \longrightarrow \mathbb{R}^{[0,1]}$

$$\alpha_i = \frac{\exp\left(\frac{q^T k_i}{\sqrt{d_v}}\right)}{\sum_{j=1}^{T} \exp\left(\frac{q^T k_j}{\sqrt{d_v}}\right)} \quad ; \quad 0 \le \alpha_i \le 1, \quad \sum_{i=1}^{T} \alpha_i = 1$$

4.) define Attention :

$$\text{Attention}(q, K, V) = AV = \sum_{i=1}^{T} \alpha_i v_i = z \in \mathbb{R}^{T \times d_v}$$

Attn $(\ ) : X^{T \times d_m} \longrightarrow Z^{T \times d_v}$ ← better representation with contextual info

Attention # query tokens :

$q \in \mathbb{R}^{T \times d_v}$

$$\text{Attention}(q, k, V) = \left[\text{Softmax}\left(\frac{q k^T}{\sqrt{d_v}}\right)\right] V$$

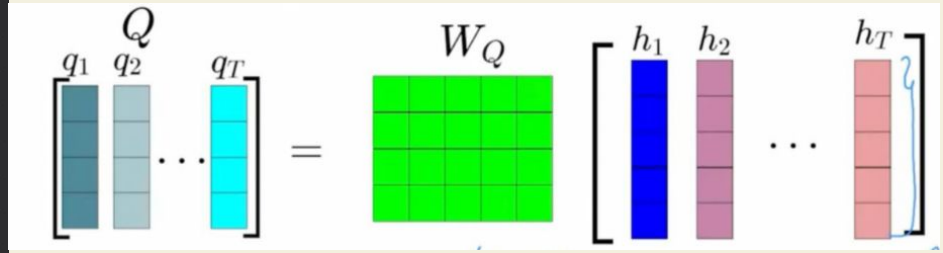
The goal

However, now both the vectors (s_i) and (h_j), for all i, j are available for all the time (whereas in the seq2seq model, h_j for all j were available and s_i was obtained one at a time)

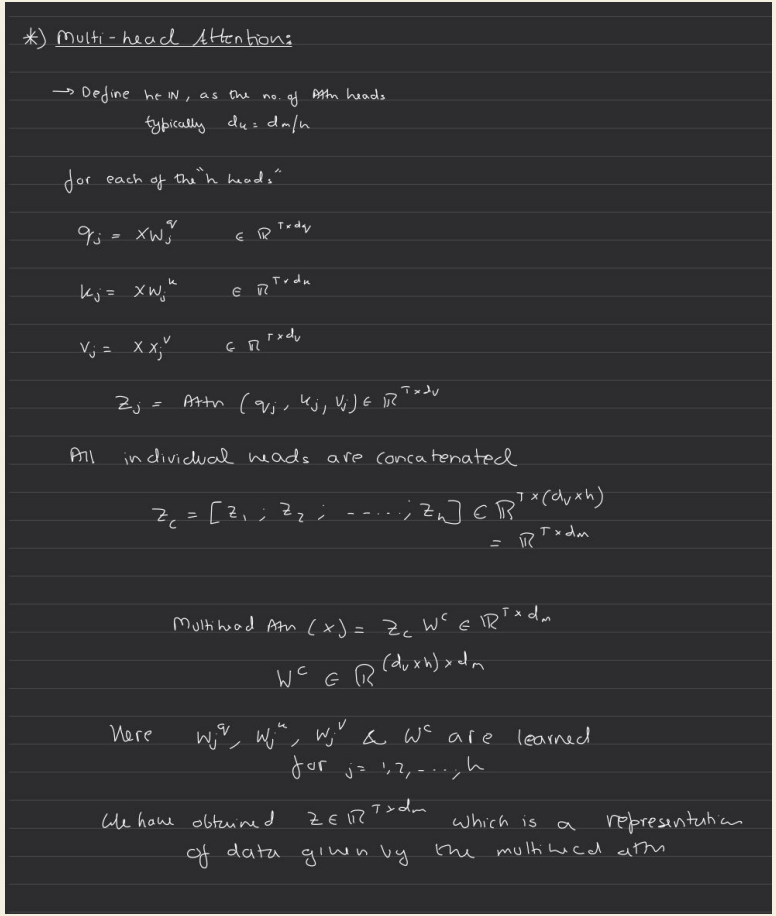
h_j

s_i	The	animal	didn't	cross	the	street	bacause	it
The	0.6	0.1	0.05	0.05	0.02	0.02	0.02	0.1
animal	0.02	0.5	0.06	0.15	0.02	0.05	0.01	0.12
didn't	0.01	0.35	0.45	0.1	0.01	0.02	0.01	0.03
cross	.							
the	.							
street	.							
because	.							
it	0.01	0.6	0.02	0.1	0.01	0.2	0.01	0.01

earlier ll ization not pass due to dependenc TxT



Multi-Head Attention



Why multi-head attention (vs one head)

Multiple views / subspaces leads to a more expressive power of the network. Each head linearly projects inputs into a different subspace, so different heads can detect different kinds of relations (short-range vs long-range, syntax vs semantics, exact token matches vs paraphrase signals).

Sentence (tokens):

`[The] [cat] [that] [the] [dog] [chased] [yesterday] [slept] [because] [it] [was] [tired]`

What we want the model to capture simultaneously

- **Long-range dependency:** *cat* → *slept* (subject → verb)
  Coreference: *it* → *cat*
- **Modifier attachment:** *yesterday* → *chased*

Three attention *heads* (each learns one pattern):

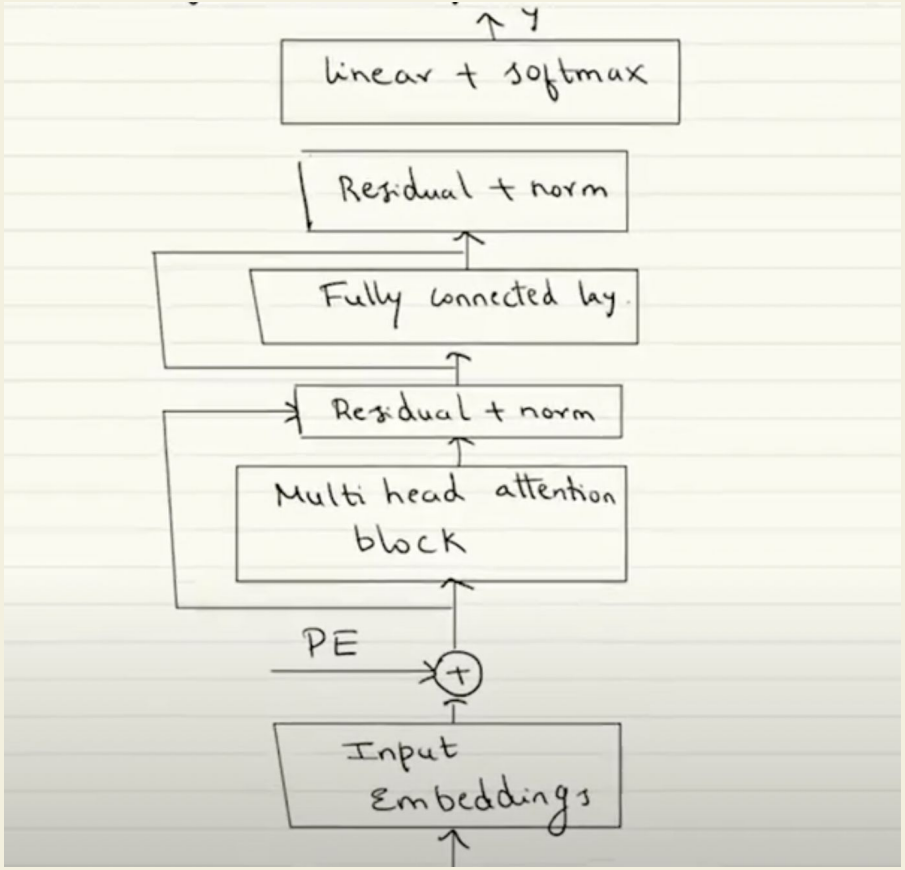
- **Head 1 (subject–verb):** Strong attention between `cat` ⇄ `slept`.
  e.g. `att(cat → slept) ≈ 0.85`, others small.
- **Head 2 (coreference):** Strong attention `it → cat` (and `cat ← it`). e.g. `att(it → cat) ≈ 0.90`.
  Head 3 (temporal modifier): `yesterday → chased`.
  e.g. `att(yesterday → chased) ≈ 0.95`.

What a single head would do A *single* attention map must combine all signals into one distribution. That often leads to a blurred map where `cat` spreads attention to `slept`, `dog`, and `it` with medium weights (e.g. 0.35, 0.25, 0.20) — none are as sharp as separate heads.

Visual intuition (arrows):

- Head 1: `cat` ⟶ `slept`
  Head 2: `it` ⟶ `cat`
  Head 3: `yesterday` ⟶ `chased`
  Single head: `cat` ⟶ `slept`, `cat` ⟶ `dog`, `cat` ⟶ `it` (all weaker / mixed)

Transformer Architecture



*) Fully connected layers:

For each of the position i in the seq, let z_i be the ith token, $z_i \in \mathbb{R}^{d_m}$

Where $\sigma(.)$ is a non-linearity

W_1 & W_2 - learnable

Output of FCL - $\hat{z}_i \in \mathbb{R}^{T \times d_m}$

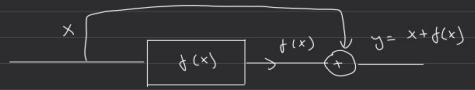
Output layer:

$$\hat{z}_j = \hat{z}_i W_p + b_p \in \mathbb{R}^V \quad ; \quad W_p \in \mathbb{R}^{d_m \times V}$$

$\hat{z}_i$ (o/p of FCL) is projected on to the space of vocab via a linear proj W_p.

$$\hat{y}_{ij} = \text{softmax}(\hat{z}_j)$$

*) Skip connections & Normalization:



Skip/Residual connection

*) Layer Normalization:

Given $x \in \mathbb{R}^{d_m}$

$$\mu = \frac{1}{d_m} \sum_{k=1}^{d_m} x_k \quad ; \quad \sigma^2 = \frac{1}{d_m} \sum_{k=1}^{d_m} (x_k - \mu)^2$$

$$\text{layernorm}(x) = \gamma \cdot \left[\frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}\right] + \beta$$

ε = small

γ & β are learnable params

if Sublayer(x) represents a block in the architecture (attn of FCL)

$$\text{output} = \text{layernorm}\left(x + \text{sublayer}(x)\right)$$

Residual connection

to ensure that the sequential ordering is preserved, an additional fixed temporal embedding vector is added to the input

positional encoding/Embedding:

→ sinusoidal Encoding:

Sps j denotes the pos in the input seq^n
& i denotes the embedding dim
then

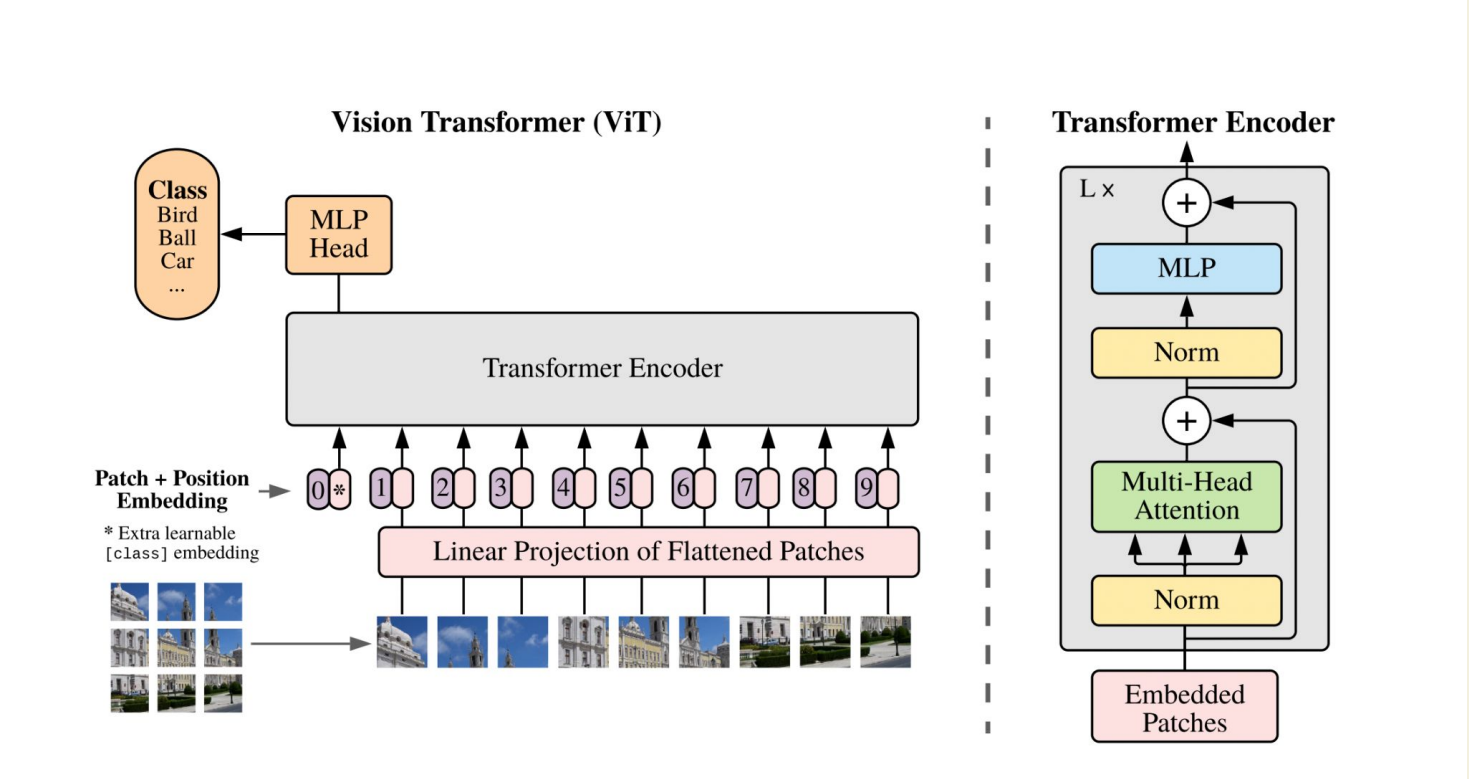
$$PE(j, 2i) = \sin\left(\frac{j}{10{,}000^{2i/dm}}\right) \qquad PE(\,) \in \mathbb{R}^{T \times dm}$$

$$PE(j, 2i+1) = \cos\left(\frac{j}{10{,}000^{2i/dm}}\right)$$

$$X_j = X_j + PE(j) \in \mathbb{R}^{dm}$$

$$\forall j \in \{1, 2, \ldots, T\}$$

Vision Transformer (ViT)



Why Transformers for Images?

Transformers are powerful for 1-D sequences (e.g., text) → self-attention enables **global context modeling** across tokens.
Vision Transformer (ViT) extends this idea to images by **reformulating an image as a sequence of tokens** (patches) and then applying a **standard Transformer encoder**.

Core motivation:

- CNNs have strong **inductive biases** (locality, translation equivariance), but these can limit scalability.
- Transformers need more data and hence are **data-hungry** but extremely **scalable** with compute and dataset size.

Minimal architectural changes from standard Transformer:

- Break image into fixed-size **patches** (e.g., 16×16).
- Flatten patches and **linearly project** them into embedding space → token sequence.
- Prepend a **learnable [CLS] token** to represent the entire image.
- Add **learnable positional embeddings** to retain spatial structure.
- Feed resulting sequence into a **stack of Transformer encoder layers**.
- Use the final [CLS] token for classification via a simple head.

Key insight: No convolutions are required — the **attention layers learn spatial relationships** directly from data.
Result: With large-scale pretraining, ViT **matches or surpasses** CNNs on image classification benchmarks.

- **Goal:** Convert an image into a **token sequence** that can be processed by a standard Transformer encoder.
- **Given:**
  - Image $x \in R^{H \times W \times C}$x
  - Patch size P×P
  - Number of patches N=H*W^P2
  - Embedding dimension D
- **Step 1: Patch Extraction**
  - Split the image into N non-overlapping patches of size P×P
  - Flatten each patch → vector of length $P2 \cdot C$
- **Step 2: Linear Projection**
  - Each flattened patch is linearly projected using $E \in R^{(P2C) \times DE}$
  - Produces **patch embeddings**: xpE∈RDxpE∈RD.
  - Add **learnable positional embeddings** $Epos \in R^{(N+1)}$ ×D to retain spatial structure.
- **Input to Transformer:**

z0=[xclass;xp1E;xp2E;…;xpNE]+Epos

Positional Embeddings

Why Positional Embeddings?

- Transformer attention is **permutation-invariant** → it doesn't know the spatial arrangement of patches by default.
- Positional embeddings inject **spatial information** into the token sequence.

Design in ViT

- ViT uses **learnable 1-D positional embeddings** $Epos \in R^{(N+1) \times D}$
  (one for each patch)
- These embeddings are **added** to patch embeddings at the input stage.

The MLP contains two layers with a GELU non-linearity.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$