* Transformer *

- Transformers are NNs designed for tasks involving sequential data like text genetion, translation, etc.

- SelfAttention:- Core mechanism allowing models to focus on imp parts of the sequence by computing relationships b/w diff words/tokens.

Architecture Breakdown:-

- Encoder-Decoder Structure:-

- Encoder:- Processes input data & creates representation.

- Decoder:- Generates output based on the encoder's representation.

- Both encoder & decoder use Multi-Head Attention layers & feed-forward Networks.

Self Attention Mechanism:-

- Given input queries (Q), keys (K) & values (V), it calculates attention scores. using the formula:-

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Scaled Dot-Product Attention:- Normalizes the dot product by dividing by $\sqrt{d_k}$ (dimension of keys) to prevent large values.

Multi-Head Attention:-

Instead of a single attention layer, transformers use multiple attention heads in parallel. The results are concatenated:

$$\text{Multi-Head}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

where each attention head is:-

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Positional Encoding:

Since transformers don't process sequences in order, positional info is added:-

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Encoder-Decoder Workflow:-

- Encoder:-
  - Input Embeddings:- Convert tokens to vectors.
  - Positional Encodings:- Embed Adds information about word order
  - Feed-Forward Net:- Adds Non-Linearity after attention layers.
  - Normalization & Residual Connections:- Stabilizes learning by normalizing & adding residual connections around attention & FFN sub-layers.

- Decoder:-
  - Works similarly to the encoder but includes:
    - Masked self-attention: Prevents future

tokens from being attended to during training.

- Encoder-Decoder Attention:- Aligns decoder output with the encoder's processed input.
- Generate sequences until stop token is encountered.

Training & Regularizeti

1- Label Smoothing:-

Instead of one-hot encoding, the target distribution is smoothed by adding some probability mass to incorrect labels, improving generalization.

$$P'(i) \quad (1-\varepsilon)\,P(i) + \frac{\varepsilon}{V}$$

2- Loss Function:-

Typically uses cross-entropy loss, incorporating KL Divergence for label smoothing.

$$L = -\sum P'(i) \log(P(i))$$

3- Optimizer:-

Uses Adam optimizer w/ learning rate scheduling:

$$lrate(step) = d_{model}^{-0.5} \cdot min(step^{-0.5}, step \cdot warmup\_steps^{-1.5})$$

Advantages Over RNNs/CNNs:-

Parallelism: Unlike RNNs that process data sequentially, transformers process all tokens simultaneously, making training much faster.

Long-Range Dependencies:-

Transformers handle distant relationships b/w tokens better due to self-attention mechanism.