**3. Methodology**

Our methodology leverages the Transformer architecture, specifically its self-attention mechanisms, to improve the accuracy and efficiency of English to Tamil translation. By focusing on careful preprocessing, model training, and evaluation, we demonstrate the efficacy of advanced deep learning techniques in language translation, particularly for low-resource languages.

**3.1 Input Tokenization and Embedding**

**3.1.1 Tokenization:** The initial processing of the input sequence involves tokenization, which breaks down the sequence into discrete units or tokens. This includes special tokens such as START\_TOKEN to denote the beginning of the sequence, text tokens in Tamil that represent the content, END\_TOKEN to mark the end of the sequence, and PAD\_TOKEN for padding shorter sequences to a uniform length. Tokenization converts these tokens into a format that can be processed by the model, resulting in a tensor representation of dimensions (30 x 50 x 512), where 30 is the batch size, 50 is the sequence length, and 512 is the embedding dimension.

**3.1.2 Embedding:** After tokenization, each token is mapped to a high-dimensional vector using an embedding layer. This layer transforms token indices into dense vectors of a specified dimension, resulting in an input tensor of dimensions (30 x 50 x 512). The embedding process captures the semantic meaning of each token and represents it in a continuous vector space, facilitating the subsequent processing stages of the Transformer model.

**3.1.3 Position Encoding:** To incorporate positional information, positional encoding is added to the token embeddings. Since the Transformer architecture does not inherently understand sequence order, positional encoding provides the model with information about the relative or absolute position of tokens in the sequence. The positional encoding is combined with the token embeddings, maintaining the tensor dimensions at (30 x 50 x 512). This allows the model to process the sequence with an understanding of token positions, which is crucial for maintaining the order and context of the sequence.

**3.2 Encoder**

The encoder processes the input sequence to generate a set of hidden representations through self-attention mechanisms. Each layer in the encoder consists of multi-head self-attention followed by a position-wise fully connected feed-forward network.

3.2.1 Feed Forward Layer: The token embeddings with positional encoding are passed through a feed-forward neural network within the encoder layer. This network comprises linear transformations followed by non-linear activation functions. The feed-forward layer transforms the embeddings by applying learned weights and biases to produce a higher-dimensional representation, enabling the model to capture complex relationships within the sequence. This transformation is crucial for the subsequent projection into Query (Q), Key (K), and Value (V) matrices.

**3.2.2 Projection into Q, K, and V Matrices:** The output of the feed-forward layer is projected into three matrices: Query (Q), Key (K), and Value (V). Each matrix has dimensions (30 x 50 x 1536), reflecting the use of multiple attention heads. For example, with three attention heads, each head might have a dimensionality of 512, resulting in a combined dimension of 1536 for the Q, K, and V matrices. These projections are essential for the self-attention mechanism, as they represent different aspects of the token representations used to compute attention scores.

**3.2.3 Splitting into Attention Heads**: The Q, K, and V matrices are split into smaller matrices for each attention head. This allows each attention head to focus on different parts of the representation space, capturing various aspects of the input sequence. For instance, each attention head’s query matrix (q) might have dimensions (30 x 50 x 64). This approach enables multi-head attention, where multiple attention mechanisms operate in parallel, enhancing the model’s ability to capture diverse patterns and dependencies within the sequence.

**3.2.4 Scaled Dot-Product Attention:** The self-attention mechanism calculates attention scores by computing the dot product of the query (q) and the transpose of the key (k) matrices, resulting in an attention score matrix of dimensions (30 x 64 x 50). The scores are scaled by the square root of the key vectors’ dimensionality and then softmax-normalized to obtain attention weights. These weights indicate the relevance of each token to every other token in the sequence. The attention weights are multiplied with the value (v) matrix to produce the output of the self-attention mechanism, reflecting the weighted combination of token representations based on their contextual relevance.

**3.2.5 Concatenation and Linear Transformation:** The outputs from the multiple attention heads are concatenated to form a single matrix. This concatenated output is then passed through a linear layer, which combines the information from all attention heads into a unified representation. The linear transformation ensures that the final attention output retains the desired dimensionality, consistent with the embedding dimension.

**3.2.6 Add & Normalize:** A residual connection is applied by adding the original input to the output of the self-attention mechanism. This residual connection helps preserve the original information and mitigates the vanishing gradient problem. After the residual addition, layer normalization is applied to the combined representation. Layer normalization standardizes the activations, improving model stability and convergence. The resulting tensor maintains dimensions of (30 x 50 x 512), ensuring consistency in the representation space.

**3.3 Decoder**

The decoder uses the encoder's hidden representations along with its own self-attention mechanisms to generate the target sequence. It incorporates an additional layer of attention over the encoder's output to focus on relevant parts of the input sentence during translation.

**3.4 Masked Multi-Head Self-Attention:** In the decoder, a multi-head self-attention mechanism is employed with masking to prevent the model from attending to future tokens during the generation of the current token. This masking ensures that each token only has access to preceding tokens, preserving the autoregressive nature of the decoder. The masked self-attention mechanism computes attention scores while adhering to this constraint, enabling left-to-right sequence generation.

**3.5 Encoder-Decoder Attention**: The decoder layer incorporates encoder-decoder attention, where the encoder's output serves as the keys and values, while the decoder's output serves as the queries. This allows the decoder to attend to relevant parts of the input sequence processed by the encoder, facilitating contextually appropriate output generation. The encoder-decoder attention mechanism aligns the decoder’s focus with the encoder’s representation, ensuring that generated tokens are informed by the input sequence.

**3.5.1 Feed Forward Network:** The combined representations from the self-attention and encoder-decoder attention mechanisms are passed through another feed-forward neural network within the decoder layer. This network applies linear transformations and non-linear activations, similar to the encoder’s feed-forward layer. This step refines the token representations before the final output projection.

**3.5.2 Add & Normalize:** Residual connections and layer normalization are applied again in the decoder layer. The residual connections add the original input to the output of the self-attention and encoder-decoder attention mechanisms, preserving input information. Layer normalization standardizes the activations, ensuring stable training and consistent representation dimensions (30 x 50 x 512).