Details of the BERT Model

(**B**idirectional **E**ncoder **R**epresentations from **T**ransformers)

By: Hiva Mohammadzadeh

Structure of the model

- Pre-trained deep learning model that uses bidirectional transformers to create context-based word embeddings.
- An encoder only model - Reads the entire sequence at once allowing the model to learn the context of a word based on all of its surrounding words.
- Consists of 12 Transformer Encoder blocks where each block has two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.
- Two versions: BERT Base and BERT Large.
- We will use BERT base with 128 tokens and an embedding dimension of 768 and 3072 feed-forward filter size.

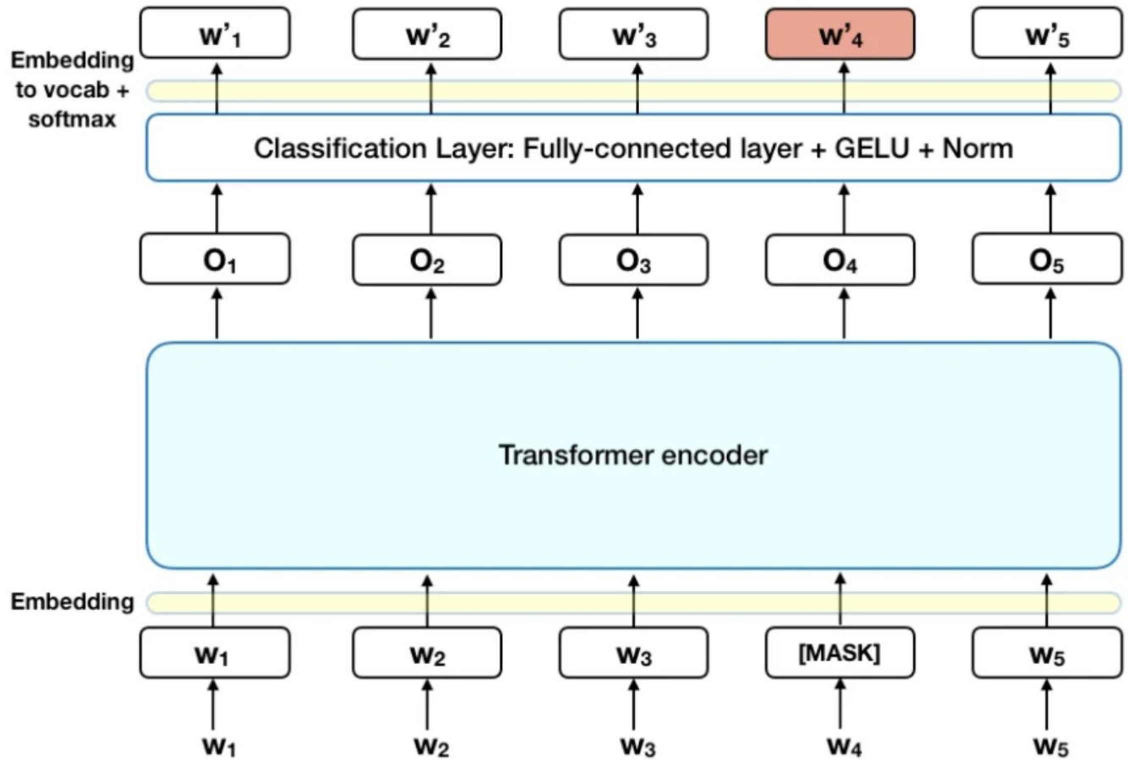
Training Scheme

- Pre-Training: Pre-trained in two stages
  a. Masked Language Model Objective:
     - Randomly mask 15% of tokens in each input sequence.
     - Train the model to predict the masked tokens based on the surrounding context.
  b. Next Sentence Prediction Objective:
     - Train the model to predict if two sentences are consecutive or not.
     - Improves the ability to handle tasks that involve multiple sentences.
- Fine-Tuning:
  a. Fine-Tune the model on specific downstream tasks by adding a task-specific output layer.

BERT Architecture

Two key ingredients:

- Transformer Encoder
- Task specific output layer

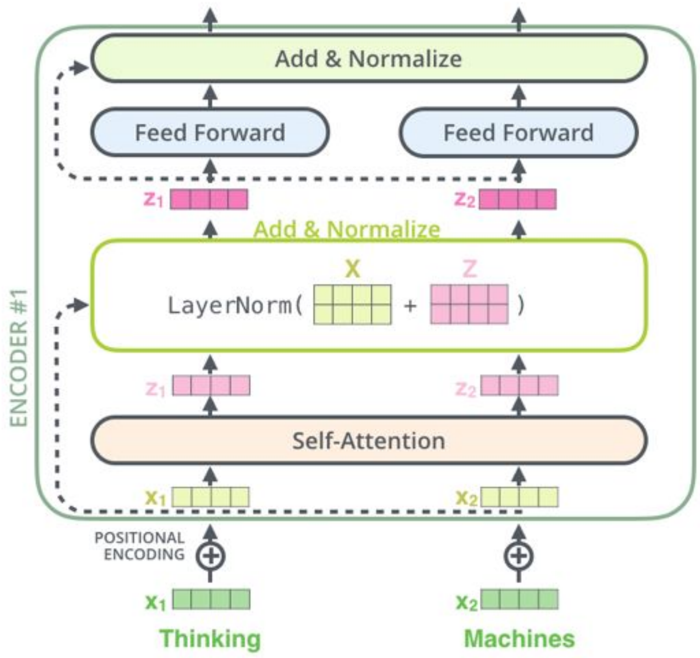


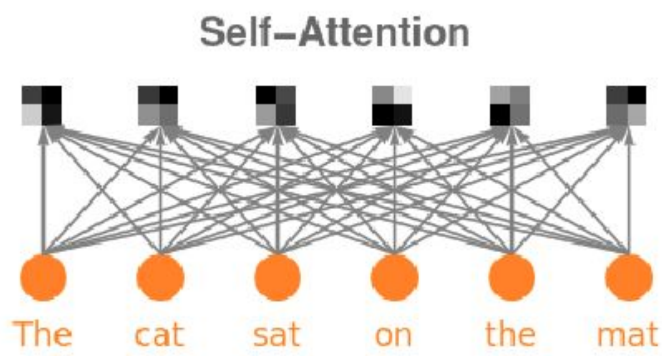
Transformer Encoder

Two components: Attention and FFN

- Self-attention is used to capture the relationships between all the tokens in the input sequence.
- Feed Forward Network (FNN) module is used in every transformer block to process the output of the normalization layer in a way to better fit it to the next attention layer.

- Residual connections are used to avoid vanishing gradient problem.
- The Layer Normalizations are used to improve the model's convergence speed.
- GELU is the non-linearity used because it has been found to perform better than the other activation functions.





Dimensions of Weight Matrices

- The input embedding matrix has a dimension of (batch_size, sequence_length, embedding_size) = (batch_size, 128, 768). But in order to do the attention parallel for different heads, we use (batch_size, 128, 768/12 heads) = (batch_size, 128, 64, 12)
- The Wk, Wv, and Wq weight matrices used in the **self-attention mechanism** have dimensions of:
    - Query weight matrix: (batch_size, hidden_size, hidden_size) = (batch_size, 768, 768)
    - Key weight matrix: (batch_size, hidden_size, hidden_size) = (batch_size, 768, 768)
    - Value weight matrix: (batch_size, hidden_size, hidden_size) = (batch_size, 768, 768)
    - Output weight matrix: (batch_size, hidden_size, hidden_size) = (batch_size, 768, 768)
- The K, V, Q matrices also have dimensions of:
    - Query matrix: (batch_size, sequence_length, hidden_size) = (batch_size, 128, 768) or (batch_size, 128, 64, 12)
    - Key matrix: (batch_size, sequence_length, hidden_size) = (batch_size, 128, 768) or (batch_size, 128, 64, 12)
    - Value matrix: (batch_size, sequence_length, hidden_size) = (batch_size, 128, 768) or (batch_size, 128, 64, 12)
- The weight matrices used in the **feed-forward neural network** have dimensions of:
    - First dense layer: (hidden_size, 4*hidden_size) = (768, 3072).
    - Second dense layer: (4*hidden_size, hidden_size) = (3072, 768).

Computation Diagram of Attention Block

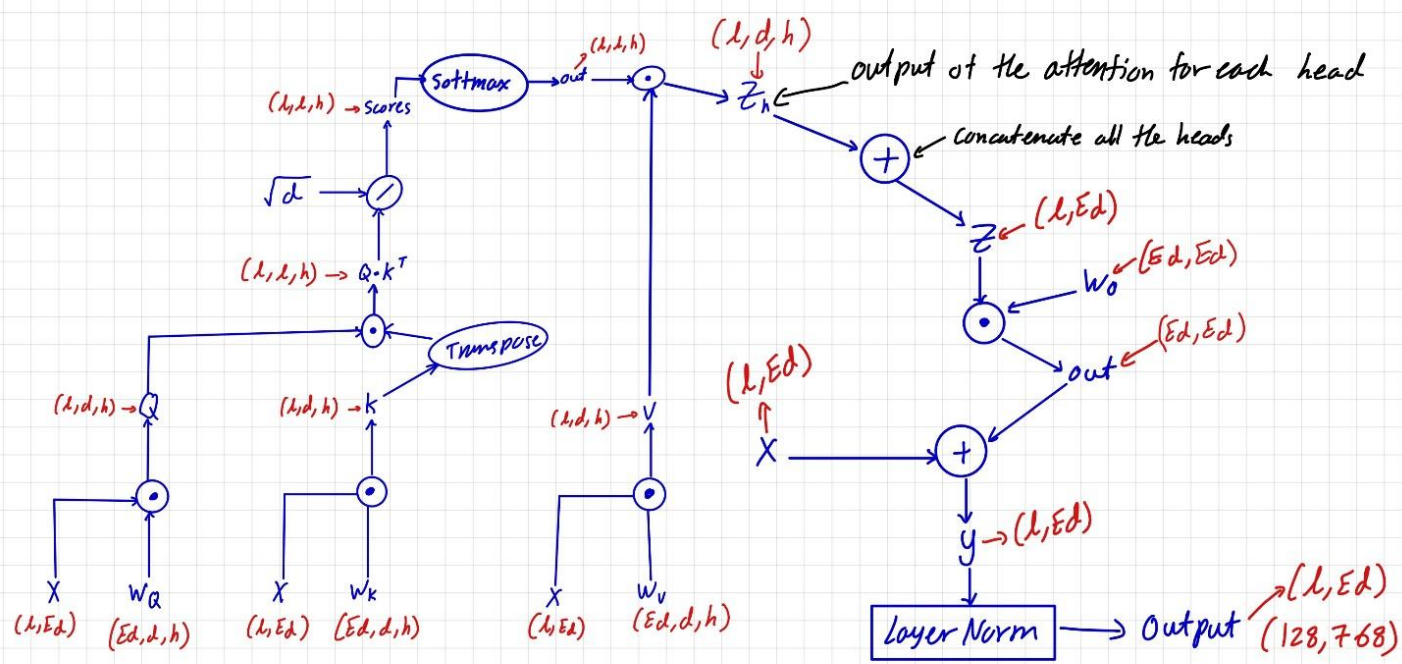
Multi-head self-Attention

Size of the embedding dimension = 768 → E_d

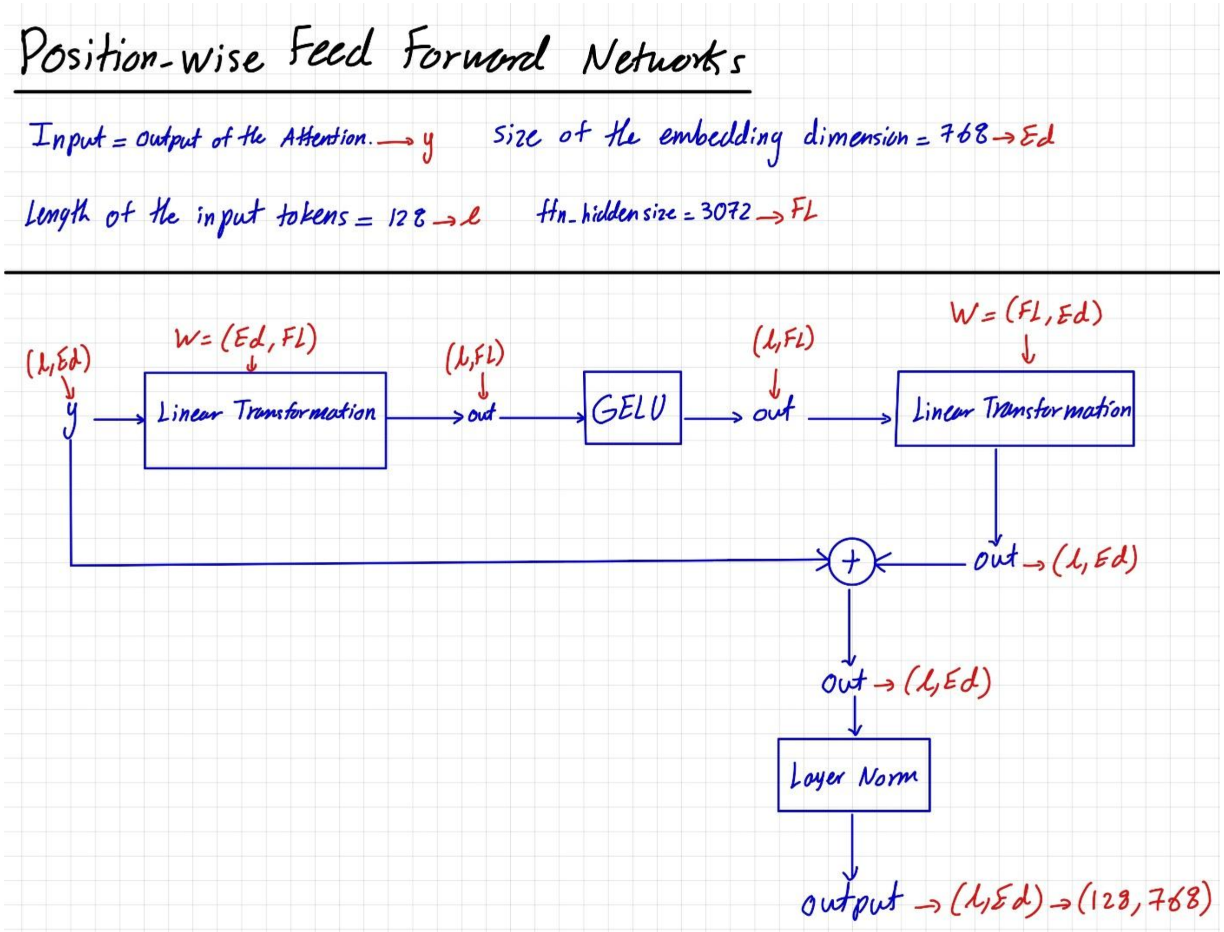
Length of the input tokens = 128 → l

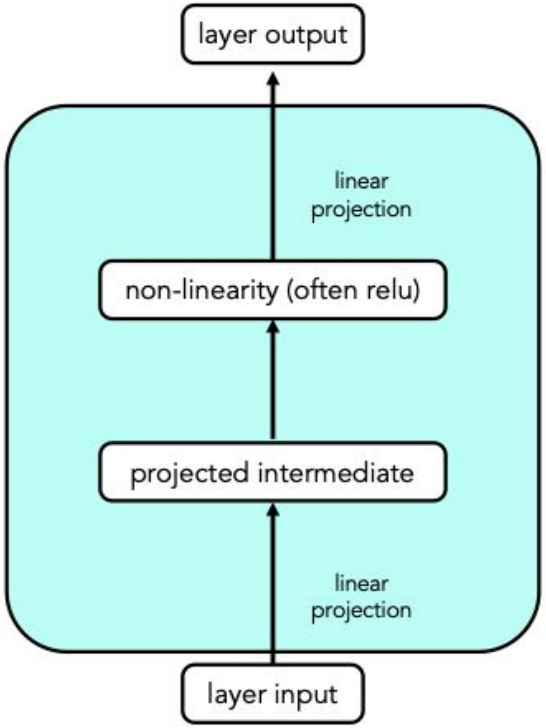
Dimension of keys, values, and queries per head $= \frac{E_d}{H} = \frac{768}{12} = 64 \rightarrow d$

Number of Heads = 12 → H

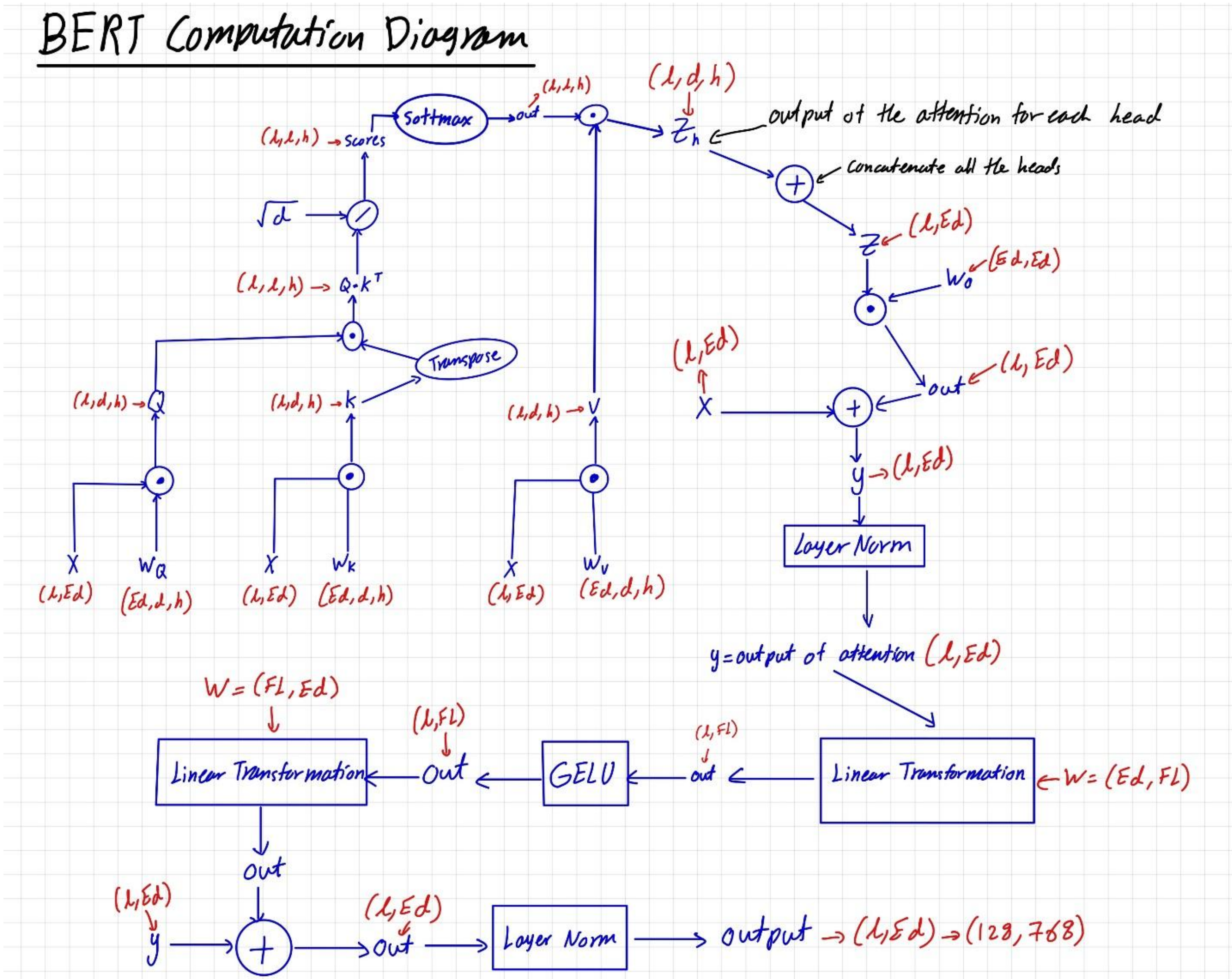


Computation Diagram of Feed Forward Layer





Computation Diagram of the whole BERT architecture



FLOPS

Assuming that every dot product requires 1 multiplications and 1 addition, I calculated the following FLOPS:

- Flops for attention block:
  - Multiplication between embedding matrix and the weight matrices of K,Q and V : 3 x 128 tokens x 768 embeddings x 768 embeddings x 2 FLOPs = 452,984,832 FLOPs
  - Multi-Head Self-Attention for 12 layers: 2 x 12 x 128 x 64 x 128 FLOPs (dot product of Q and K) + 2 x 12 x 128 x 64 x 128 FLOPs (scaled dot-product attention between output of softmax and V) + 2 x 128 x 768 x 768 FLOPs (concatenation of heads and matrix multiplication of W0) = 201,326,592
- FLOPS for Feed Forward Network Module:
  - Feed-Forward Network: FLOPs = 128 x 768 x 3072 x 2 FLOPs x 2 FLOPs + 128 FLOPs = 1,207,959,552
- Total FLOPs for 12 transformer block: FLOPs = 1,207,959,552 FLOPs (FFN) + 201,326,592 FLOPs (MHA) + 452,984,832 FLOPs (weight matrix calculations) = 1,862,270,976 FLOPs