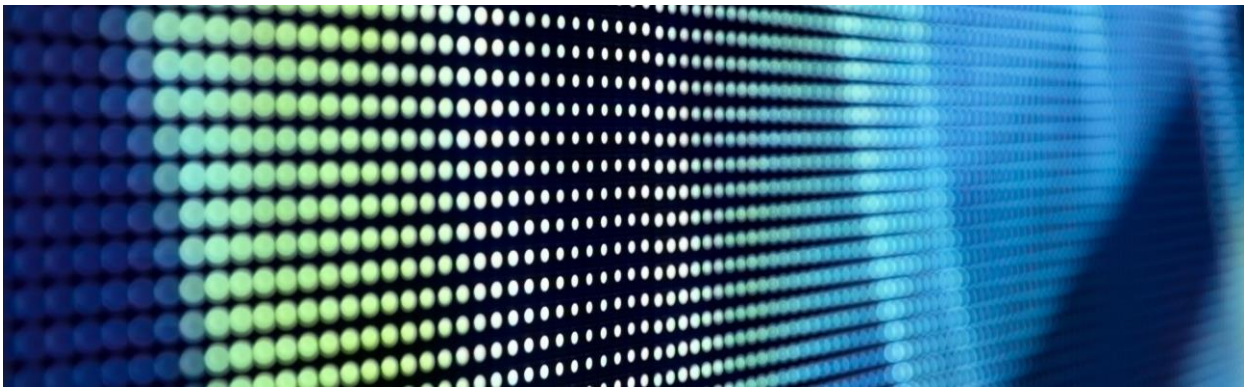

Learnings:

1 Dimension of q, k, v:

Batch size, seq length, feature model dimensions

E.g.

Batch Size: 2, meaning we are processing 2 sequences at a time.

Sequence Length: 5, indicating each sequence contains 5 elements (which could be tokens in a sentence).

Model Dimension: 64, representing that each token in the sequence is encoded by a 64-dimensional vector.

The shapes of the tensors are:

Query tensor shape: torch.Size([2, 5, 64])

Key tensor shape: torch.Size([2, 5, 64])

Value tensor shape: torch.Size([2, 5, 64])

2 Reason for scale the dot products by 1/sqrt(d_k):

We suspect that for large values of d_k , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients.

3 Environment configuration for cuda to run on the local machine

4 Transformer Architecture:

Implementing the Transformer model enhanced understanding of its architecture, including self-attention mechanisms and positional encoding.

5 PyTorch Usage:

Working with PyTorch for implementing deep learning models provided hands-on experience.

Challenges Faced:

1

Understanding Transformer Architecture: Initially challenging but overcome by referencing research papers and breaking down components.

2

Debugging: Resolving tensor dimensions, masking, and loss computation errors using print statements, shape inspection, and step-by-step execution.

3

Hyperparameter Tuning: Essential for optimization, required multiple iterations and experimentation to find optimal settings.

Implementation Process:

The model architecture included components like multi-head attention, position-wise feedforward networks, positional encoding, encoder layers, decoder layers, and the overall Transformer architecture. The code was structured into modular components for easy understanding and modification.