1 Question 1

The square mask is used to ensure that when processing sequences (such as in language modeling or sequence generation tasks), the transformer does not attend to future tokens during training. This is important for tasks like autoregressive text generation, where the model should only consider past and current tokens while predicting the next one.
Transformers do not take position into account because they do not use recurrence (like RNNs) or convolutions. Instead, they rely on self-attention, which treats all tokens equally without any notion of order. To incorporate the information about the relative positions of tokens in a sequence, positional encodings are added to the input embeddings.

2 Question 2

We replace the classification head because classification tasks require predicting a single class label, whereas language modeling generates a sequence of token probabilities. The classification head maps the transformer's output to class scores.
The key difference between language modeling and classification is that language modeling predicts the next token in a sequence, outputting probabilities for each token, while classification assigns a single label to the entire input, making it a global prediction task.

3 Question 3

3.1 Language Modeling Task

The total number of trainable parameters is the sum of parameters from the embedding layer, the transformer layers, and the language modeling head.
- Embedding Layer: The embedding layer has a weight matrix of size ntoken $\times$ nhid, where ntoken is the vocabulary size and nhid is the hidden dimension.

$$\text{Embedding Parameters} = \text{ntoken} \times \text{nhid}$$

- Transformer Layers: Each transformer encoder layer consists of multi-head self-attention and feed-forward networks. Let:
- nhead be the number of attention heads,
- nhid be the hidden dimension (which is equal to the model dimension),
- nlayers be the number of transformer layers.

For each transformer layer, the parameters include:
1. Self-Attention Parameters: Each head computes three matrices: query, key, and value. Each matrix has size nhid $\times$ (nhid/nhead), and we have 3 matrices per head and nhead heads, so:

$$\text{Self-Attention Parameters (per layer)} = 3 \times (\text{nhid} \times \frac{\text{nhid}}{\text{nhead}}) \times \text{nhead} = 3 \times \text{nhid}^2$$

2. Feed-Forward Network Parameters: The feed-forward network consists of two layers, with parameters of size nhid $\times$ nhid:

$$\text{FFN Parameters (per layer)} = 2 \times (\text{nhid} \times \text{nhid})$$

Thus, the total parameters per transformer layer are:

$$\text{Transformer Parameters (per layer)} = 3 \times \text{nhid}^2 + 2 \times \text{nhid}^2 = 5 \times \text{nhid}^2$$

For nlayers layers, we have:

$$\text{Total Transformer Parameters} = \text{nlayers} \times 5 \times \text{nhid}^2$$

- Language Modeling Head: The size of the weight matrix is nhid $\times$ ntoken.

$$\text{Language Modeling Head Parameters} = \text{nhid} \times \text{ntoken}$$

Therefore, the total number of parameters for the language modeling task is:

$$\text{Total Parameters (LM)} = \text{ntoken} \times \text{nhid} + \text{nlayers} \times 5 \times \text{nhid}^2 + \text{nhid} \times \text{ntoken}$$

3.2 Classification Task

For classification, the number of trainable parameters is similar to the language modeling task, except for the classification head, which replaces the language modeling head.
- Embedding Layer: Same as in language modeling.

$$\text{Embedding Parameters} = \text{ntoken} \times \text{nhid}$$

- Transformer Layers: Same as in language modeling.

$$\text{Total Transformer Parameters} = \text{nlayers} \times 5 \times \text{nhid}^2$$

- Classification Head: The classification head maps the final hidden state to the class scores. If there are nclasses classes, the weight matrix size is nhid $\times$ nclasses.

$$\text{Classification Head Parameters} = \text{nhid} \times \text{nclasses}$$

Therefore, the total number of parameters for the classification task is:

$$\text{Total Parameters (Classification)} = \text{ntoken} \times \text{nhid} + \text{nlayers} \times 5 \times \text{nhid}^2 + \text{nhid} \times \text{nclasses}$$

4 Question 4

The plot illustrates the validation accuracy of two models during training: one trained from scratch and the other pretrained. Key observations include the pretrained model consistently outperforming the scratch model, indicating that its initial weights, learned from a larger dataset, provide a robust foundation for the task. Both models exhibit early convergence, plateauing in validation accuracy around 10 epochs, suggesting they effectively learn the validation data patterns. While both models achieve generally high validation accuracy, the pretrained model's superior performance emphasizes the advantages of transfer learning with better stability.

5 Question 5

One of the limitations of the language modeling objective used in the notebook, compared to the masked language model objective introduced in BERT, is that the former typically processes text in a unidirectional manner (left-to-right or right-to-left), which restricts the model's ability to utilize context from both sides of a word or phrase. In contrast, BERT's masked language model objective allows for bidirectional context by randomly masking words in the input and training the model to predict them based on the surrounding context from both directions. This bidirectional approach enables BERT to capture more nuanced relationships and dependencies within the text, leading to improved performance on various natural language processing tasks.

References