Question 1

The square mask ensures the model does not access future tokens during training, maintaining an autoregressive structure. This prevents the model from ”cheating” by looking ahead.
Positional encoding is used to give the model a sense of token order, since transformers process sequences in parallel. Sinusoidal values are added to token embeddings, encoding the relative positions of tokens.

Question 2

The classification head is replaced because the output structure differs between tasks. In language modeling, we predict the next token for each position, generating a probability distribution over the vocabulary. In classification, a single label is predicted for the entire sequence. Thus, the classification task requires a different head to output class labels instead of tokens.

Question 3

For both tasks:

- **Embedding layer:** The embedding layer has $V \times d$ trainable parameters, where V is the vocabulary size and d is the hidden dimension (nhid).
- **Transformer Encoder:** Each transformer encoder layer contains:
  - Multi-head attention: Each head has $3 \times d \times d_k$ parameters for the query, key, and value projections, where $d_k = d/n_{\text{heads}}$. Since there are n_{heads} heads, the total number of parameters for the multi-head attention is:
    $$3 \times (d \times d_k) \times n_{\text{heads}} = 3 \times d^2$$
    Additionally, the output projection matrix W_O has $d_{\text{model}} \times d$ parameters, adding another d^2 parameters. Therefore, the total for multi-head attention is:
    $$4 \times d^2$$
  - Feedforward layers: Each transformer encoder layer includes a feedforward network with two linear layers: one with $d \times d_{\text{ff}}$ parameters and another with $d_{\text{ff}} \times d$. If $d_{\text{ff}} = d$, then the total number of parameters for the feedforward layers is:
    $$2 \times d^2$$
  - Layer normalization and biases: Each encoder layer has two layer normalization operations (before the attention mechanism and before the feedforward layers), each with $2 \times d$ parameters (scale and bias), resulting in:
    $$4 \times d$$

  The total number of parameters for each transformer encoder layer is:
  $$4 \times d^2 + 2 \times d^2 + 4 \times d = 6 \times d^2 + 4 \times d$$
  For n_{layers} transformer encoder layers, the total is:
  $$n_{\text{layers}} \times (6 \times d^2 + 4 \times d)$$

Language modeling task:

- The output layer for language modeling is a linear layer mapping from d to V (the size of the vocabulary), so it adds $d \times V$ parameters.

- Total number of trainable parameters:

$$V \times d + n_{\text{layers}} \times (6 \times d^2 + 4 \times d) + d \times V = 2 \times V \times d + n_{\text{layers}} \times (6 \times d^2 + 4 \times d) = 1003200$$

Classification task:

- The classification head is a linear layer that maps from d to n_{classes}, which adds $d \times n_{\text{classes}}$ parameters.
- Total number of trainable parameters:

$$V \times d + n_{\text{layers}} \times (6 \times d^2 + 4 \times d) + d \times n_{\text{classes}} = 964000$$

Question 4

- The pretrained model shows significantly higher validation accuracy from the beginning of the training process. It starts at about 80% accuracy after the first epoch, and fluctuates slightly but remaining close to 80% in subsequent epochs. This indicates that pretraining gives the model a strong initial representation, allowing it to quickly reach a high level of performance and generalize better.
- The model trained from scratch begins with a lower accuracy (around 60%) and improves more gradually over the first few epochs, reaching a plateau around 75% after the fifth epoch. Its validation accuracy slightly fluctuates but remains lower than the pretrained model.
- **Conclusion:** Pretraining provides a clear advantage in terms of both faster convergence and better generalization, as evidenced by the consistently higher validation accuracy of the pretrained model. This demonstrates the effectiveness of pretraining for improving performance in downstream tasks.

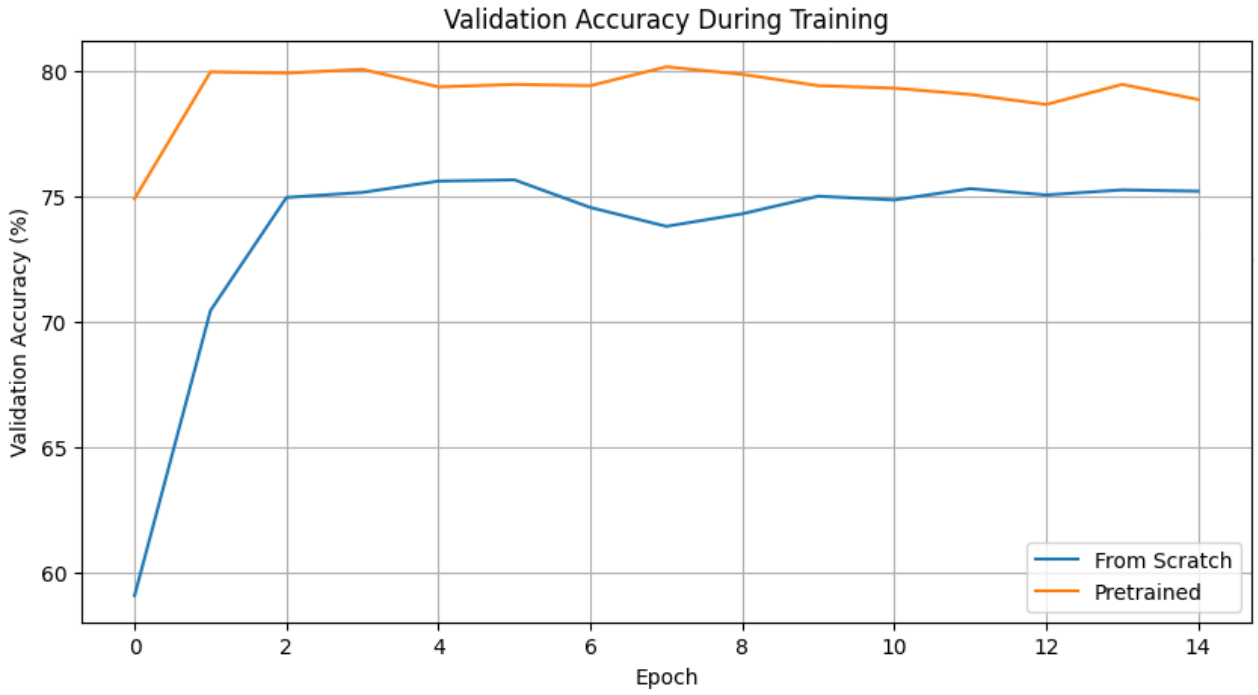

Figure 1: Validation Accuracy During Training

Question 5

The main limitation of the standard language modeling objective, compared to BERT's masked language model (MLM) objective, is that it is **unidirectional**. In traditional language models, the model can only predict tokens from left-to-right or right-to-left, meaning that it can only attend to past tokens when predicting the next word. This prevents the model from fully capturing bidirectional context.
In contrast, the MLM objective introduced in BERT allows the model to predict masked tokens by leveraging both **past and future context**. This is achieved by randomly masking some tokens in the input and training

the model to predict them based on the full sequence. As a result, BERT's model is **bidirectional**, leading to richer contextual representations, as highlighted in the paper: "a deep bidirectional model is strictly more powerful than either a left-to-right model or a shallow concatenation of a left-to-right and a right-to-left model" (BERT, Devlin et al. 2018). Therefore, the MLM objective enables the model to better capture the nuances of natural language.