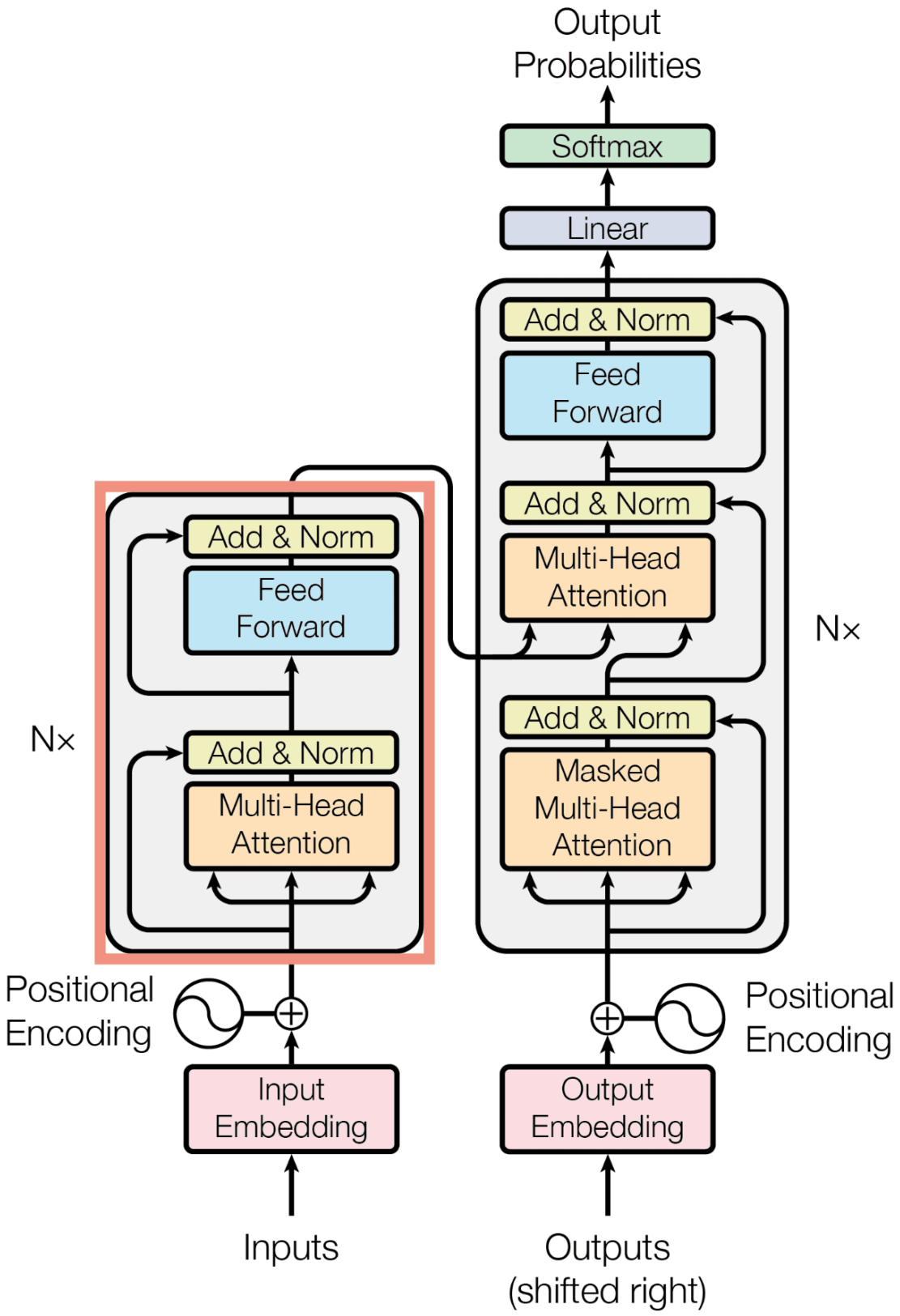
Nisarg Bhavsar, Zaid Ahmed Khan
21CH30009, 21CH30067

BTP Project

13th September 2024

Prof. Ayantika Chatterjee

Piecewise Training of a Transformer



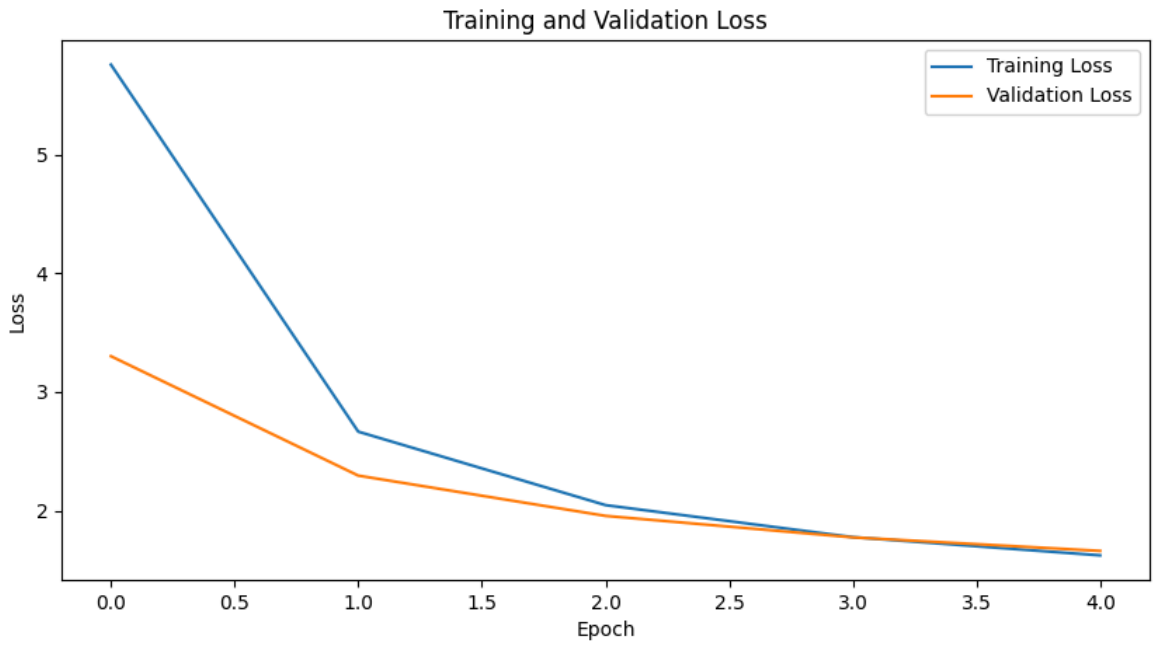
A: Only Embedding Model

```python
class EmbeddingModel(nn.Module):
    def __init__(self, vocab_size, d_model):
        super().__init__()
        self.vocab_size = vocab_size
        self.d_model = d_model
        self.embedding = nn.Embedding(vocab_size, d_model)
        self.output_layer = nn.Linear(d_model, vocab_size)

    def forward(self, x):
        x = self.embedding(x)
        return self.output_layer(x)

    def generate_text(self, x):
        x = self.forward(x)
        return x.argmax(dim=-1)
```

We got the following results when training an only embedding model:



This model had the worst loss value after 5 epochs of training. (~2.0)

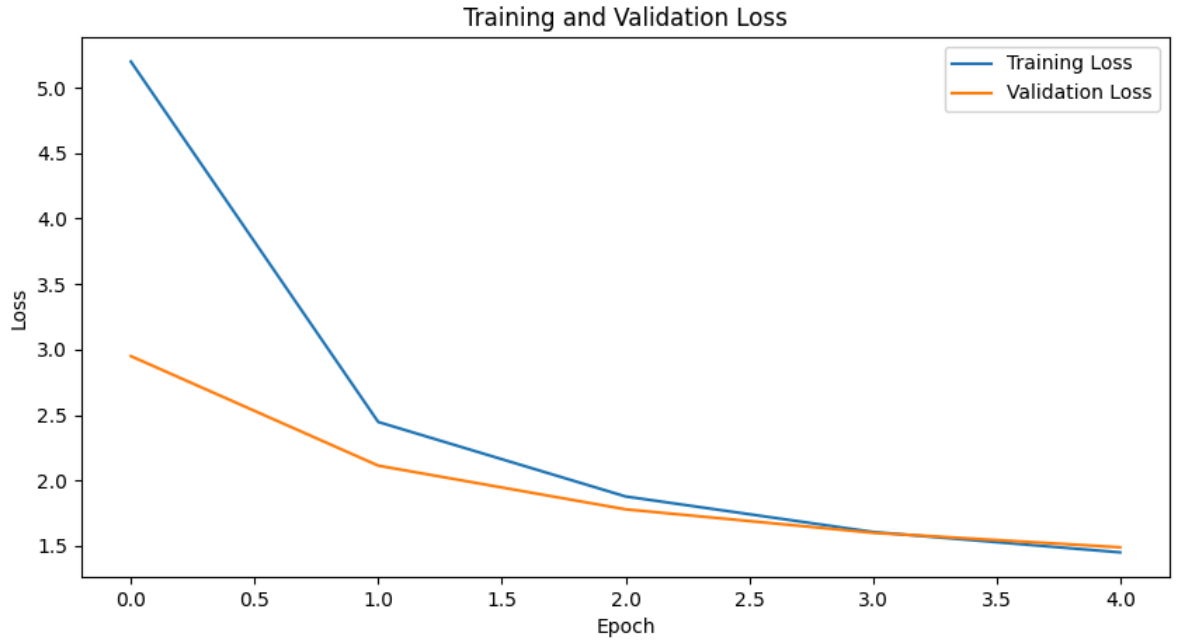
B: Positional Embedding Model

```python
class PositionalModel(nn.Module):
    def __init__(self, vocab_size, d_model):
        super().__init__()
        self.vocab_size = vocab_size
        self.d_model = d_model
        self.embedding = nn.Embedding(vocab_size, d_model)
        self.positional_encoding = PositionalEncoding(d_model)
        self.output_layer = nn.Linear(d_model, vocab_size)

    def forward(self, x):
        x = self.embedding(x)
        x = self.positional_encoding(x)
        return self.output_layer(x)

    def generate_text(self, x):
        x = self.forward(x)
        return x.argmax(dim=-1)
```

We got the following results when training a positional embedding model:



This model had a better loss value after 5 epochs of training. (~1.5)

C: Attention Model

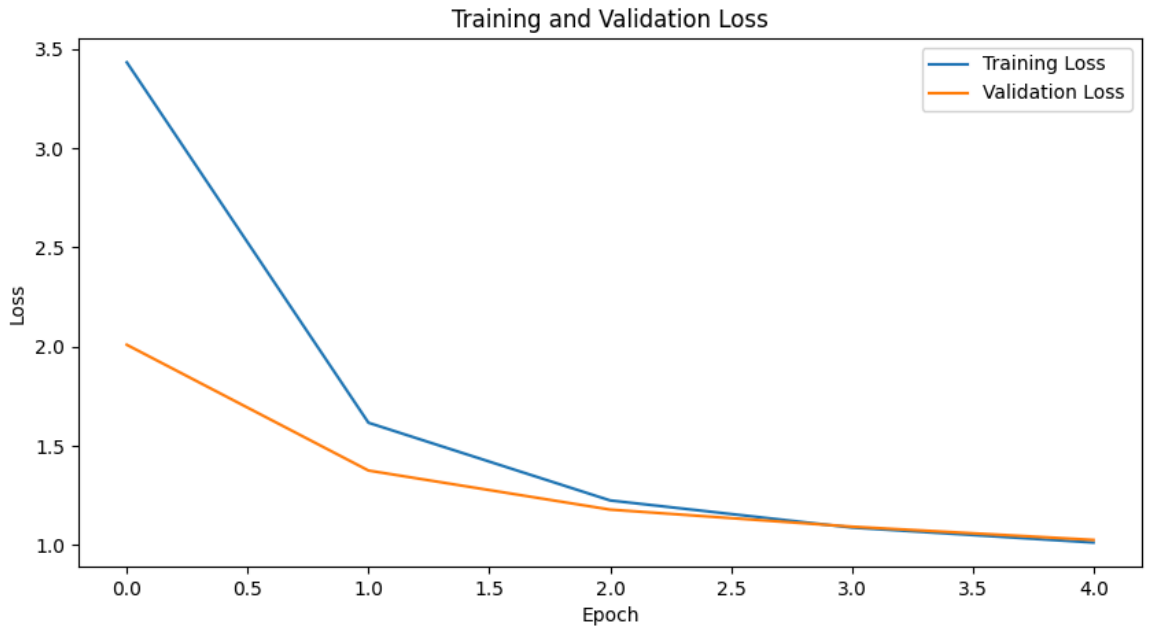
```python
class DecoderModel(nn.Module):
    def __init__(self, vocab_size, d_model, N, heads):
        super().__init__()
        # Input embedding layer
        self.vocab_size = vocab_size
        self.input_embedding = nn.Embedding(vocab_size, d_model)
        self.positional_encoding = PositionalEncoding(d_model)
        self.decoder_layers = nn.ModuleList([DecoderLayer(d_model, heads) for _ in range(N)])
        self.output_layer = nn.Linear(d_model, vocab_size)

    def forward(self, x, mask=None):
        # Apply input embeddings
        x = self.input_embedding(x)
        x = self.positional_encoding(x)
        for decoder_layer in self.decoder_layers:
            x = decoder_layer(x, mask)
        x = self.output_layer(x)

        return x

    def generate_text(self, x):
        x = self.forward(x)
        return x.argmax(dim=-1)
```

We got the following results when training an attention model:



This model had the best loss value after 5 epochs of training. (~1.0)