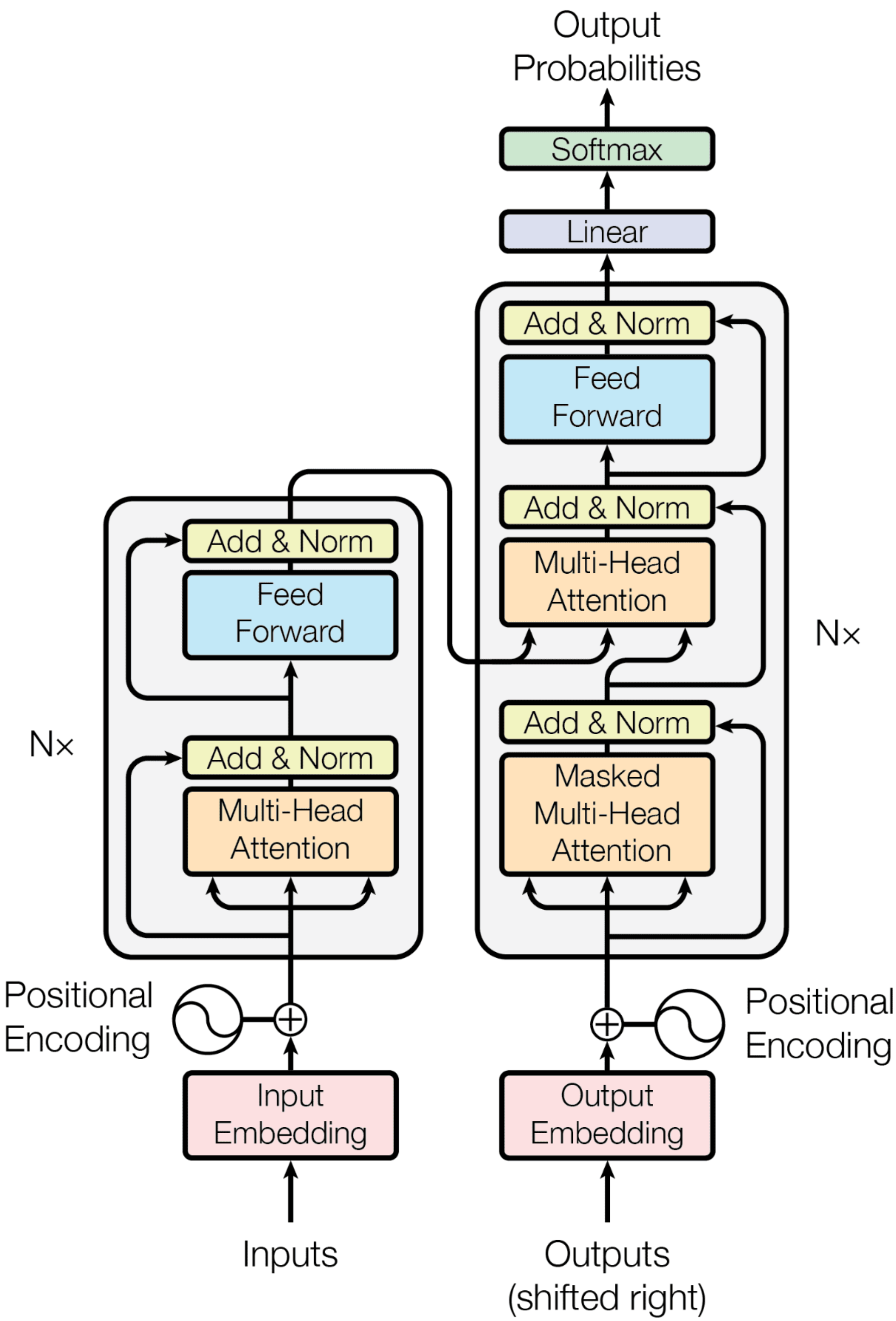
Part 6 :


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

1. What are the key components of the Transformer architecture?

- Multi-head self-attention
- Positional encoding
- Feed-forward networks
- Layer normalization
- Residual connections

---

2. Why is positional encoding used in Transformers?

- Transformers do not have inherent sequential processing like RNNs.
- Positional encoding provides information about the relative positions of tokens.
- Helps the model understand the order of words in a sequence.

---

3. What is the role of the multi-head attention mechanism?

- Allows the model to focus on different parts of the input sequence simultaneously.
- Captures multiple relationships between tokens at different levels.
- Provides more expressive power by combining attention scores from multiple heads.

---

4. How do residual connections improve the Transformer architecture?

- Helps mitigate the vanishing gradient problem.
- Speeds up convergence by allowing gradients to flow more easily through the network.
- Adds the input back to the output, facilitating better learning.

---

5. What is the difference between scaled dot-product attention and traditional attention?

- Scaled dot-product attention divides the dot product by the square root of the dimension of the key vectors ($\sqrt{d_k}$).
- This scaling ensures more stable gradients and prevents large values from destabilizing the softmax function.
- Traditional attention does not use this scaling factor, which can lead to gradient issues.

---

6. How is the feed-forward layer in Transformers implemented?

- It is a two-layer fully connected neural network.
- Each token in the sequence is processed independently.
- Typically uses ReLU or GELU activation functions between the layers.

---

7. What is the function of layer normalization in Transformers?

- Normalizes the output of each layer to maintain consistent mean and variance.
- Stabilizes training and accelerates convergence.
- Reduces the risk of exploding or vanishing gradients during training.

---

8. What are the roles of the encoder and decoder in a Transformer?

- **Encoder:** Encodes input sequences into context-rich representations.
- **Decoder:** Generates output sequences by attending to encoder outputs and previously generated tokens.
- **Encoder-decoder attention:** Enables the decoder to focus on relevant parts of the input sequence during generation.

---

9. How does self-attention differ between the encoder and decoder in Transformers?

- **Encoder self-attention:** Attends to all tokens in the input sequence.
- **Decoder self-attention:** Attends to tokens generated so far and masks future tokens to prevent data leakage.
- **Encoder-decoder attention:** In the decoder, attends to encoder outputs while generating tokens.

---

10. Why is scaling (dividing by √d_k) used in attention mechanisms?

- Prevents large dot-product values from distorting the softmax function.
- Ensures more stable gradients during backpropagation.
- Helps avoid issues with exploding gradients, especially with large input dimensions.

1. How does learning rate scheduling optimize the training process of generative models over time?

- **Dynamic Adjustment:** Learning rate scheduling adjusts the learning rate during training to improve convergence and prevent overshooting the optimal solution.
- **Warmup:** A gradual increase in the learning rate during initial epochs helps stabilize training and prevent divergence.
- **Decay Strategies:** Common techniques like exponential decay or cosine annealing reduce the learning rate over time, ensuring finer updates as the model approaches convergence.
- **Prevention of Overfitting:** By lowering the learning rate in later stages, the model avoids oscillations and overfitting.

---

2. Explain transfer learning in the context of natural language processing (NLP). How do pre-trained language models contribute to various NLP tasks?

- **Definition:** Transfer learning in NLP involves leveraging pre-trained models (e.g., BERT, GPT) trained on large text corpora for specific downstream tasks.
- **Feature Extraction:** Pre-trained models provide contextual embeddings that capture syntax and semantics.
- **Fine-Tuning:** Models are fine-tuned on task-specific datasets (e.g., sentiment analysis, question answering) for improved performance.
- **Advantages:** Saves computational resources, reduces the need for large labeled datasets, and achieves state-of-the-art results on various NLP tasks.

---

3. Compare and contrast GPT and BERT.

Feature	GPT	BERT
Architecture	Decoder-only Transformer	Encoder-only Transformer
Training	Autoregressive (predict next token)	Masked Language Model (predict masked tokens)
Context	Unidirectional (left-to-right)	Bidirectional (both directions)
Usage	Text generation	Text understanding tasks

- **Conclusion:** GPT excels at generative tasks, while BERT is optimized for understanding-based tasks like classification and NER.

---

4. What issues in RNNs are addressed by transformer models?

- **Sequential Processing:** RNNs process data sequentially, causing inefficiency; transformers process input in parallel.
- **Vanishing Gradient:** Long-term dependencies in RNNs are hard to learn due to vanishing gradients; transformers use attention mechanisms to capture dependencies effectively.
- **Fixed Context Length:** RNNs struggle with long sequences; transformers handle long-term dependencies better with global attention.

---

5. What distinguishes transformers from RNNs and LSTMs?

- **Parallelism:** Transformers process sequences in parallel, unlike the sequential nature of RNNs/LSTMs.
- **Attention Mechanism:** Transformers use self-attention to model dependencies, while RNNs rely on recurrent connections.
- **Scalability:** Transformers scale better with large datasets due to parallel processing and are more computationally efficient for long sequences.

---

6. How does BERT work, and what makes it unique compared to traditional NLP models?

- **Bidirectional Context:** BERT learns from both left and right context, unlike traditional unidirectional models.
- **Pre-training Tasks:** Includes Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).
- **Transfer Learning:** Fine-tunes pre-trained representations for downstream tasks, achieving state-of-the-art performance.
- **Transformer Encoder:** Relies on transformer encoders for deep contextual understanding.

---

7. Why is incorporating relative positional information critical in transformer models?

- **Importance:** Positional encoding allows the model to understand the order of tokens in sequences, critical for context.
- **Relative Encoding:** Captures relationships between tokens irrespective of their absolute position, beneficial for tasks like music generation or time-series prediction.
- **Example:** In machine translation, understanding word order impacts grammar and meaning.

---

8. What challenges arise from the fixed attention span in the vanilla Transformer model?

- **Limited Context:** Fixed-length input restricts the model's ability to process long documents or sequences.
- **Dependency Loss:** Important information outside the attention window may be ignored, reducing accuracy for long-term dependencies.

9. Why isn't simply increasing context length a viable solution for handling longer contexts in transformers?

- **Computational Complexity:** Self-attention scales quadratically with input length (O(n2)), making it memory-intensive.

  O(n2)O(n^2)

- **Hardware Limitations:** Larger context sizes require extensive GPU/TPU resources.
- **Solution:** Techniques like sparse attention, Reformer, or Longformer mitigate these issues.

10. Can you explain how self-attention works in transformer models?

- **Key, Query, Value (KQV):** Each token generates these vectors.
- **Attention Score:** Compute similarity between query and keys using dot product, followed by softmax.
- **Weighted Summation:** Multiply attention scores with value vectors to form the output.
- **Parallel Computation:** Enables relationships between tokens across the sequence to be modeled efficiently.

11. What pre-training mechanisms are typically used in Large Language Models (LLMs)?

- **Masked Language Modeling (MLM):** BERT-like models predict randomly masked tokens in a sentence.

- **Autoregressive Modeling:** GPT-like models predict the next token in a sequence.
- **Seq2Seq Modeling:** Models like T5 combine encoder-decoder architecture for both generation and understanding tasks.

---

12. Why is multi-head attention essential in transformer models?

- **Diversity in Representation:** Captures different types of relationships between tokens (e.g., syntax and semantics).
- **Efficiency:** Enables the model to focus on various parts of the sequence simultaneously.
- **Improved Learning:** Aggregates insights from multiple subspaces, enhancing overall context understanding.