GenAI

AI

ML , NLP , STT & TTS, Robotic

SL, USL and **RL**

Classification, Regression Clustering

DL

NN

ANN , CNN and GRU

Deep Neural Networks • Architectures

- Artificial Neural Networks(ANN)
  - Recurrent Neural Networks(RNN)
    - Long Short-Term Memory(LSTM) - TimeSeries
- Convolutional Neural Networks (CNN) - Images
- Gated Recurrent Units (GRU)

Actual Problem ?

Before the advent of transformers, traditional neural network architectures for sequence processing, such as recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), were commonly used. While these architectures showed success in certain tasks, they had limitations that made them less effective in capturing long-range dependencies and managing sequential information.

- **Sequential Processing Limitations:**
- RNNs and LSTMs process sequences sequentially, which can be computationally inefficient and limits their ability to capture long-range dependencies. This becomes a significant drawback, especially in tasks involving long sequences, like natural language processing.
- **Vanishing and Exploding Gradients:**
- RNNs can suffer from vanishing and exploding gradient problems during training, which makes it challenging for them to learn dependencies over long sequences. LSTMs were introduced to mitigate some of these issues but still had limitations.
- **Parallelization**:
- Traditional sequence models like RNNs are inherently sequential, making parallelization during training difficult. This limits their ability to take full advantage of modern GPU architectures for faster training.

Transformers :

Transformers, in the context of machine learning, refer to a type of neural network architecture introduced in the paper **"Attention is All You Need"** by Vaswani et al. in 2017.

This architecture is especially powerful for **sequence-to-sequence tasks**, where input and output are both sequences, such as language translation, text summarisation, and more.

a. Self-Attention Mechanism (scaled dot-product attention)

b. Encoder and Decoder Architecture

c. Attention Heads and Layers

d. Postion Encoding

Use Cases:

- **Natural Language Processing (NLP):** Transformers have excelled in NLP tasks, including machine translation, sentiment analysis, text summarization, and question answering.
- **Speech Recognition:** Transformers can be applied to process and understand speech signals.

- **Image Processing:** While initially developed for NLP, transformers have been adapted for computer vision tasks, such as image classification and object detection. Vision Transformer (ViT) is an example.
- **Recommendation Systems:** Transformers can be used to model complex user-item interactions in recommendation systems.
- **Drug Discovery:** Transformers are employed in the analysis of molecular data for drug discovery.
- **Code Generation:** Transformers can generate code snippets based on natural language descriptions.

Examples :

1. BERT (Bidirectional Encoder Representations from Transformers). - Google
    a. RoBERTa (Robustly optimized BERT approach)
    b. DistilBERT
    c. ALBERT (A Lite BERT)
2. GPT (Generative Pretrained Transformer) —— OPenAI

L

T5 -Google

Dive into the concepts of Encoder & Decoder

1. Encoder

A. Input Embedding:

- Each word/token in the input sequence is first converted into a dense vector using token embeddings.

- Positional Encoding: Since Transformer doesn't have a built-in notion of sequence order, a positional encoding is added to each embedding to provide positional information.

B. Self Attention Mechanism:

- Queries (Q), Keys (K), and Values (V) are derived from the input embeddings.
- Attention scores are computed by taking the dot product of Q and K, followed by scaling and applying a softmax.
- The scores determine how much focus each word in the sequence should have on every other word.
- The attention outputs are computed by multiplying the attention scores with V.

C. Feed-Forward Neural Network:

- Each attention output is passed through a feed-forward neural network (identical for each position).
- This is followed by layer normalization and a residual connection.

D. Stacking:

- Multiple such encoder layers (usually 6 or more in models like BERT and the original Transformer) are stacked to form the encoder part.

2. Decoder

A. Input Embedding:

- Just like the encoder, the decoder also starts with embedding the input tokens (which are the output tokens so far) and adding positional encodings.

B. Self Attention Mechanism:

- The mechanism is similar to the encoder's self attention, but with one major difference: to maintain auto-regressive property, positions in the decoder can only attend to earlier positions in the output sequence.
- This is achieved using a masked version of the attention to prevent future tokens from being used.

C. Encoder-Decoder Attention Layer:

- After the masked self-attention layer, the decoder has another attention mechanism.
- This attention layer helps the decoder focus on relevant parts of the input sentence, similar to how attention works in seq2seq models with LSTMs.
- Here, the Q comes from the decoder's previous layer, and K and V come from the encoder's output.

-

D. Feed-Forward Neural Network:

- Just like in the encoder, the output of the encoder-decoder attention goes through a feed-forward network.

E. Stacking:

- Again, multiple decoder layers (usually the same number as encoder layers) are stacked to form the complete decoder.

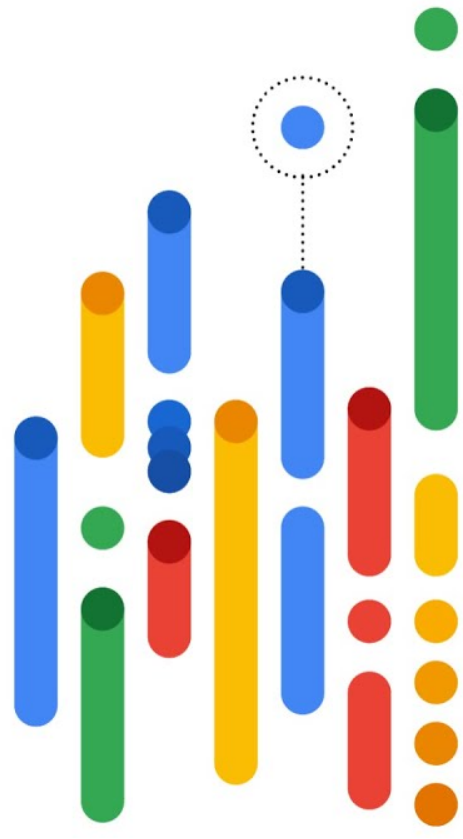
Reference Links

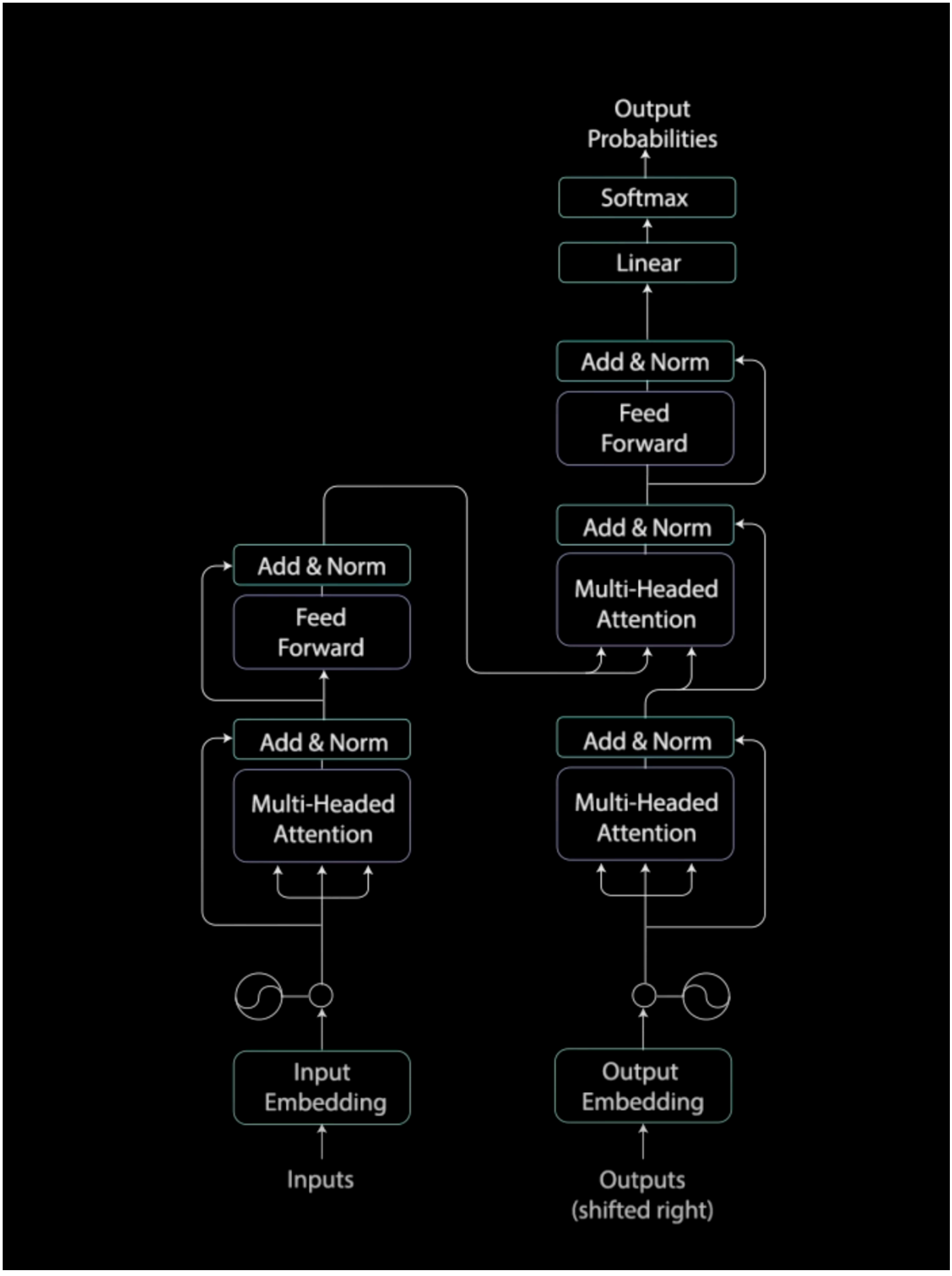
https://cedar.buffalo.edu/~srihari/CSE676/12.4.7 TransformerModels.pdf
https://github.com/tugot17/Vision-Transformer-Presentation
https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0
https://towardsdatascience.com/drawing-the-transformer-network-from-scratch-part-1-9269ed9a2c5e
https://towardsdatascience.com/transformers-explained-65454c0f3fa7
https://towardsdatascience.com/transformers-explained-65454c0f3fa7
https://ketanhdoshi.github.io/Transformers-Arch/
https://daleonai.com/transformers-explained
https://github.com/Hannibal046/Awesome-LLM
https://arxiv.org/pdf/1810.04805.pdf

YT

Google Cloud

Transformer Models and BERT Model: Overview




Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Headed Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Headed Attention
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)