Transformers in AI and Generative AI

Introduction

Transformers are a family of neural network architectures designed to process sequences by learning relationships between elements through attention. They transformed natural language processing and generative AI by replacing recurrence and convolution with attention-based computation that scales efficiently to long contexts. This document explains transformers in depth, shows concrete numerical and conceptual examples, and gives practical notes for using and visualizing transformers in a 4-page Word document.

Overview and Motivation

- **Why transformers were created** Traditional sequence models used recurrent networks which processed tokens one at a time and struggled with long-range dependencies and parallelization. Transformers enable parallel processing of tokens and let the model dynamically weight interactions among all tokens in a sequence, solving long-range dependency problems and enabling large-scale training.
- **Key high-level ideas**
  - **Parallelism** through simultaneous token processing.
  - **Attention** as a learned weighting of relationships between tokens.
  - **Modularity** via stacked layers of attention and feed-forward blocks.
  - **Scalability** that permits billions of parameters enabling rich generative behavior.

Transformer Architecture

Core components

- **Input Embedding** Tokens (words, subwords, pixels, patches) are converted to dense vectors called embeddings. Positional information is added so the model knows token order.
- **Encoder and Decoder Blocks** Original architecture has an **encoder** stack that creates contextualized token representations and a **decoder** stack that produces outputs autoregressively in generation tasks. Modern variants sometimes use only encoder (for classification) or only decoder (for text generation).
- **Layer structure** Each block typically contains:
  - **Multi-Head Self-Attention** that lets each token attend to others using several parallel attention heads.
  - **Positionwise Feed-Forward Network** which applies a small MLP to each token independently.
  - **Residual Connections** that add input to output to ease gradient flow.
  - **Layer Normalization** applied to stabilize and accelerate training.

Transformers in AI/GenAI

1. Introduction

Transformers are neural network aeficapittdy architectures used in artificial intelligence (AI) and generative AI (GenAI). It enables machines to process and generate human-like text, images, and other data.

Transformers are desigued to handle sequential data by focusing on the relationship between different elements in a sequence, aligning in a network architecture.

2. The Transformer Architecture

Transformers are composed of an encoder and a decoder. The encoder processes the input data and the decoder generates output Transformers use it to process.

- Multi-head attention and [illegible] neural networks.
- For-ted - fornerd [illegible]

Inputs are represted as embeddings.

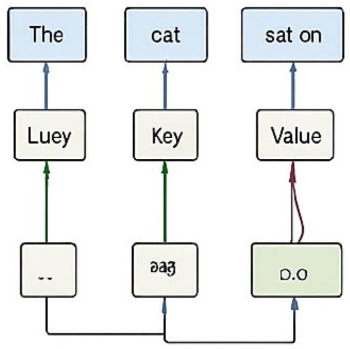

Figure 2: Self-Attention Mechaism

4. Applications of Transformers

Transformers art uied in a various tangus-[illegible], lnoage generation, and text

3. Self-Attention Mechanism

The transformer mechanism allows transgen ders to weighthe importance of ditferent word in a sentence when generating output.

The mecaniars compéur oai seaie a seore representing the importance of each word s in relation to others.

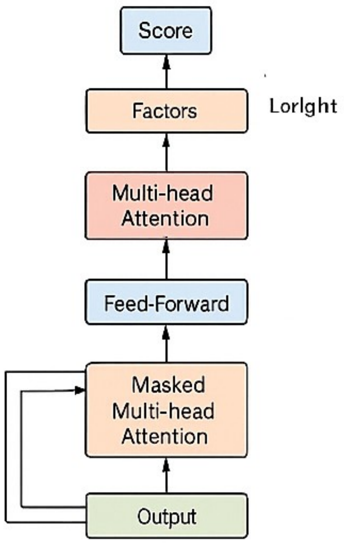

Figure 2: Self-Attention Mechoru

4. Applications of Transformers

Transformers are used in a various tasks such as language translation, image generation, and text summarization.

In language translation, transformers (con dense long documents into shorter summaries.

In image generation, əcantrcanformers is like DALL-F-dhag based on textual descri-

tnrips outmage generation transformers based on tex.ruũral description, actuns

Data flow example

- Input sentence: **"The quick brown fox jumps"**
    1. Tokenize to tokens.
    2. Map tokens to embeddings and add positional encodings.
    3. Pass through N stacked encoder layers; each layer refines contextual representations.
    4. For generation, decoder uses masked self-attention to prevent peeking at future tokens and attends to encoder outputs to condition on input.

Multi-Head Attention intuition

- Each attention head learns a distinct type of relationship such as syntax, short-range dependencies, or coreference. Combining multiple heads gives the model richer relational representation.

Self-Attention Mechanism Deep Dive

Query Key Value idea

- For each token, the model computes three vectors: **Query (Q)**, **Key (K)**, and **Value (V)** by linear projections of the token embedding.
- Attention scores are dot-products of Q with all K vectors, scaled, turned into probabilities via softmax, and used to weight the V vectors to produce the token's context-aware output.

Numerical example

- Suppose token A has Q_A, and tokens B and C have K_B, K_C and V_B, V_C.
    - Compute scores sB = Q_A · K_B, sC = Q_A · K_C.
    - Apply scale and softmax to get weights wB, wC.
    - Output for token A = wB * V_B + wC * V_C. Concrete values help: if sB = 2, sC = 1 with softmax → wB ≈ 0.73, wC ≈ 0.27, so A mostly copies information from B's value.

Multi-head formulation

- Instead of a single QKV set, the model uses H heads with smaller dimensions each. Outputs of all heads are concatenated and projected to form the final attended vector for the layer.

Masked attention in decoding

- During generation, the decoder uses a causal mask so attention only considers previous tokens, ensuring the model cannot condition on future tokens.

Complexity and scaling

- Attention computes pairwise interactions across T tokens leading to O(T^2) memory and compute per layer. This explains practical design choices for input length and motivates sparse or efficient attention variants for very long inputs.

Training, Objectives, and Variants

Typical training objectives

- **Autoregressive language modeling**: model predicts next token given previous tokens (used by GPT-style decoders).
- **Masked language modeling**: model predicts masked tokens from context (used by BERT-style encoders).
- **Seq2Seq objectives**: encoder-decoder models trained to map input sequences to output sequences (translation, summarization).

Optimization and regularization

- Large-scale training uses Adam-style optimizers, learning-rate schedules (warmup + decay), dropout, and weight decay. Residual connections and normalization help train deep stacks.

Notable variants and adaptations

- **Encoder-only** (BERT) for understanding tasks.
- **Decoder-only** (GPT) for generative tasks.
- **Encoder-decoder** (T5, original Transformer) for sequence-to-sequence tasks.
- **Vision Transformers (ViT)** split images into patches treated like tokens and apply transformer layers.
- **Sparse and Long-Range Transformers** modify attention to handle very long sequences efficiently.

Transfer learning and fine-tuning

- Pretraining on huge corpora followed by supervised fine-tuning on downstream tasks is standard. Fine-tuning may adapt entire model weights or add lightweight adapters for parameter-efficient tuning.

Applications and Examples

Natural Language Processing

- **Machine Translation** Example: English sentence “She loves coffee” encoded and decoded into French “Elle aime le café.” The encoder learns contextual embeddings; the decoder attends to those embeddings to produce a fluent translation.
- **Summarization** Long input document encoded; decoder generates a concise summary by attending to salient encoder representations.
- **Question Answering** Model attends over passage tokens to locate answers; attention highlights relevant spans.

Generative AI beyond text

- **Image Generation** Transformers generate or autoregressively decode image tokens; text-to-image models map text embeddings to image tokens or latent representations.
- **Audio and Music** Sequence modeling for waveform or symbolic music generation using attention over temporal tokens.
- **Multimodal models** Transformers combine text, image, and other modalities by aligning token streams through cross-attention.

Concrete walkthrough for a translation task

1. Input: "The cat sat on the mat" → tokens + positional encodings.
2. Encoder produces context vectors where "mat" influences representation of "sat" through attention.
3. Decoder masked self-attends to previously generated French tokens and cross-attends to encoder outputs to produce next token probabilities.
4. Greedy or beam search decodes final sentence "Le chat s'est assis sur le tapis" (example of beam search improving fluency).

Practical Notes and Suggested Diagrams for the Word Document

What to include visually

- **Figure 1 Architecture Diagram**: input token embeddings → stacked encoder layers → decoder stack → softmax output.
- **Figure 2 Self-Attention Flow**: show token Q, K, V vectors, dot-product scores, softmax weights, weighted sum to produce output.
- **Figure 3 Multi-Head Attention**: parallel heads with different colored lines and concatenation.
- **Figure 4 Example Attention Map**: heatmap showing how the word "it" attends to "the dog" earlier in the sentence.
- **Figure 5 Applications Collage**: small panels showing translation, summarization, image generation, and code generation.

Writing and formatting tips for 4 pages

- Page 1: Title, Introduction, Overview and Motivation, start Architecture.
- Page 2: Finish Architecture, Self-Attention numeric example with small equations and a figure.
- Page 3: Training and Variants, transfer learning, practical optimization notes.
- Page 4: Applications, concrete walkthroughs, recommended diagrams, conclusion and further reading pointers.

Suggested captions and alt text

- Provide concise captions under each figure describing what the diagram illustrates and a short alt text for accessibility describing main visual elements.

Conclusion

Transformers replaced sequential computation with attention-driven, parallelizable layers that learn contextual relationships across tokens. Their flexibility supports encoder, decoder, and hybrid models applied to language, vision, audio, and multimodal tasks. Including diagrams of the architecture, attention flow, and attention maps will make the 4-page Word document clear and instructional for readers at both conceptual and practical levels.