Foundational Model Architecture in Generative AI: Components & Use Cases

EURON

scl

Contents

Decoder-Only Architecture

✅ Overview:

- Uses only the **decoder** part of the Transformer architecture.
- Processes input autoregressively, meaning tokens are generated one at a time.
- Uses self-attention and causal masking (to prevent looking at future tokens).

◆ Examples:

- **GPT models** (GPT-3, GPT-4, LLaMA, Falcon, etc.)
- **ChatGPT**
- **Claude**

◆ How It Works:

1. Takes input tokens and processes them using **self-attention**.
2. Generates output **token by token** in an **autoregressive** manner.
3. Uses **causal masking** to ensure each token only attends to previous tokens.

◆ Strengths:

✅ **Good for open-ended generation**: Ideal for text generation, dialogue systems, and story writing.

✅ **Efficient inference**: Since there's no encoder, it requires less computation than encoder-decoder models for text generation.

✅ **Pretrained on large-scale corpora**: Good at capturing **long-term dependencies** and generating fluent, coherent text.

◆ Weaknesses:

❌ **Struggles with bidirectional understanding**: Lacks a strong understanding of sentence structure since it doesn't process full context at once.

❌ **Less efficient for document summarization**: Since it generates token by token, it's less structured compared to an encoder-decoder model.

❌ **Poor at structured outputs**: Not ideal for tasks requiring structured outputs, like translation or question answering.

◆ Best Use Cases:

- **Chatbots (ChatGPT, Claude, LLaMA)**
- **Story or text generation**
- **Code generation (Codex, StarCoder, etc.)**

- **Autoregressive NLP tasks (completion, rewriting, summarization in some cases)**

Encoder-Decoder (Seq2Seq) Architecture

✅ Overview:

- Consists of two components: **encoder** and **decoder**.
- Encoder processes the input as a whole (bidirectional).
- Decoder generates output based on the encoder's representation.

◆ Examples:

- **T5 (Text-to-Text Transfer Transformer)**
- **BART**
- **mT5 (Multilingual T5)**
- **NLLB (No Language Left Behind)**
- **Whisper (Speech-to-Text)**

◆ How It Works:

1. **Encoder**:
   - Processes the entire input at once.
   - Uses **self-attention** (bidirectional) to understand context.
2. **Decoder**:
   - Takes the encoded representation as input.
   - Uses **cross-attention** with self-attention to generate output.
   - Produces output token by token (autoregressive).

◆ Strengths:

✅ **Bidirectional understanding**: The encoder captures **context from the entire input** (better comprehension).
✅ **Great for structured outputs**: Works well for **translation, summarization, and question-answering**.
✅ **Handles long inputs better**: Can compress and summarize large texts efficiently.

◆ Weaknesses:

❌ **Slower inference**: Requires passing input through both an **encoder and decoder**, making it computationally expensive.

❌ **More complex training**: Needs **paired datasets** (input-output mappings like English-French for translation).

◆ **Best Use Cases:**

- **Machine translation (Google Translate, NLLB)**
- **Text summarization (T5, BART)**
- **Question answering (T5, UL2)**
- **Text-to-text NLP tasks (T5-based models)**

Key Differences:

Feature	**Decoder-Only (GPT, LLaMA, Falcon)**	**Encoder-Decoder (T5, BART, NLLB)**
Architecture	Uses only a **decoder**	Uses both **encoder & decoder**
Processing	Autoregressive (token-by-token)	Encoder processes input first, then decoder generates output
Attention	**Self-attention only**, causal masking	**Self-attention (encoder), cross-attention (decoder)**
Bidirectional Context	❌ No	✅ Yes
Inference Speed	🔥 Faster (one-pass decoding)	🐢 Slower (two-pass encoding & decoding)
Memory Usage	✅ Lower (no encoder)	❌ Higher (requires both encoder & decoder)
Tasks Best Suited	Text generation, Chatbots, Code generation	Summarization, Translation, Question Answering
Examples	GPT-3, GPT-4, ChatGPT, Claude, Falcon	T5, BART, mT5, Whisper, NLLB

Which One to Choose?

Use Case	**Best Model**
Chatbot, Creative Text Generation	✅ **Decoder-Only (GPT, ChatGPT, Claude, LLaMA)**
Summarization	✅ **Encoder-Decoder (T5, BART, UL2)**
Machine Translation	✅ **Encoder-Decoder (mT5, NLLB, MarianMT)**
Speech-to-Text	✅ **Encoder-Decoder (Whisper)**

Use Case	Best Model
Code Generation	✅ **Decoder-Only (Codex, StarCoder, CodeLLaMA)**
Question Answering (structured)	✅ **Encoder-Decoder (T5, UL2)**

Hybrid Approaches

Some modern models combine elements of both architectures:

- **T5 (Text-to-Text Transfer Transformer)**: Encoder-Decoder but reformulates all NLP tasks as text-to-text.
- **UL2 (Unifying Language Learning)**: Can act as both **decoder-only** and **encoder-decoder**, adapting based on task.
- **Gemini**: Uses a mix of encoder-decoder processing for different modalities.

GPT-3.5 Architecture

GPT-3.5 is an intermediate model between GPT-3 and GPT-4, offering improvements in efficiency, cost, and response quality.

◆ Architecture Breakdown

- **Base Model:** Transformer-based architecture (decoder-only)
- **Number of Parameters:** ~175B (GPT-3 based assumption, OpenAI hasn't disclosed exact figures)
- **Layers:** ~96 transformer layers
- **Hidden Dimension (d_model):** 12,288
- **Attention Heads:** 96 heads with 128-dimensional embeddings each
- **Vocabulary Size:** ~50,000+ BPE tokens
- **Context Length:** 4K tokens (Extended to 16K in GPT-3.5-turbo)

◆ Key Components

1. **Tokenization:** Uses Byte Pair Encoding (BPE) to tokenize input.
2. **Input Embeddings:** Converts tokens into dense vectors.
3. **Positional Encoding:** Rotary Position Embeddings (RoPE) for better long-range dependencies.
4. **Multi-Head Self-Attention (MHSA):** Computes token dependencies.
5. **Feedforward Network (FFN):** Processes contextualized embeddings.
6. **Layer Normalization:** Normalizes activations for stability.
7. **Dropout & Regularization:** Used to prevent overfitting.

8. **Logits & Softmax:** Produces probabilities over the vocabulary.

◆ Training Pipeline

- **Pretraining:** Trained on a mixture of internet text (~45TB)
- **Training Method:** Causal Language Modeling (CLM) with next-token prediction
- **Optimization:** AdamW optimizer with gradient checkpointing
- **Loss Function:** Cross-entropy loss
- **Scaling Laws:** Model follows OpenAI's scaling laws for efficient performance

---

GPT-4 Architecture

GPT-4 significantly improves upon GPT-3.5 in terms of multimodal capabilities, efficiency, and reasoning.

◆ Architecture Breakdown

- **Base Model:** Transformer-based, but with enhancements
- **Parameters:** Estimated 1T+ (OpenAI hasn't disclosed exact numbers)
- **Layers:** ~120+ transformer layers
- **Hidden Dimension (d_model):** ~16,384 (speculated)
- **Attention Heads:** ~128 heads
- **Context Length:** 32K tokens (4x GPT-3.5)
- **Training Data:** More diverse, including code, books, and multimodal data

◆ Major Enhancements

1. **Mixture of Experts (MoE) (Speculative)**
   - GPT-4 is likely a Mixture of Experts model (used in PaLM and GLaM).
   - Instead of activating all parameters, it activates only subsets of neurons.
   - **Benefits:** Lower computational cost, higher efficiency.
2. **Multimodal Capabilities**
   - Accepts both text and images.
   - Vision transformer (ViT) may be integrated for image processing.
3. **Improved Positional Encoding**
   - Likely uses **ALiBi (Attention Linear Bias) or Rotary Positional Embeddings (RoPE)**.
   - Handles longer context windows efficiently.
4. **Optimized Training**
   - **Distributed Training:** Uses TPUv4 or A100/H100 GPUs.
   - **Parallelism Strategies:** ZeRO, FSDP, and tensor/model parallelism.
   - **Gradient Checkpointing:** Reduces memory footprint.
5. **Improved Fine-Tuning & Alignment**

- Reinforcement Learning from Human Feedback (RLHF).
- Constitutional AI (rule-based reinforcement).
- Safety optimizations to minimize biases.

GPT-4 Turbo Architecture

GPT-4 Turbo is optimized for speed and cost efficiency while maintaining high performance.

◆ Architecture Breakdown

- **Base Model:** Enhanced GPT-4 architecture
- **Efficiency:** Optimized with custom GPU hardware (Possibly OpenAI's in-house silicon)
- **Context Length:** 128K tokens (Largest in OpenAI models)
- **Training Data:** More up-to-date, possibly extending to 2024

◆ Key Improvements

1. **Low Latency Inference**
    - Optimized KV-cache for faster inference.
    - Smart batching with dynamic compute allocation.
2. **Memory-Efficient Attention Mechanisms**
    - **FlashAttention** for reducing computational overhead.
    - **Sparse Attention** to focus on relevant tokens dynamically.
3. **Mixture of Experts (MoE) (Highly Probable)**
    - Routes queries to only active sub-networks.
    - **Outcome:** Model runs at lower cost while maintaining intelligence.
4. **Hardware Optimization**
    - Designed to run on **OpenAI's custom chips**.
    - **Better parallelism** than previous GPT models.
5. **Fine-Tuned for Cost-Effective Deployment**
    - **Lower training and inference costs** using improved quantization.
    - **Optimized distillation techniques**.

Step-by-Step Overview of GPT Model Flow

Input Preprocessing

- **User Input:** Tokenized using BPE.
- **Token Embeddings:** Mapped to dense vectors.
- **Positional Encoding:** Injects sequence order.

Forward Pass in Transformer

- **Multi-Head Self-Attention (MHSA)**
  - Splits input into multiple attention heads.
  - Computes query-key-value (QKV) matrices.
  - Aggregates weighted attention scores.
- **Feedforward Network (FFN)**
  - Applies non-linearity (ReLU/GELU).
  - Processes token representations.
- **Layer Normalization**
  - Ensures numerical stability.
- **Dropout Regularization**
  - Prevents overfitting.

Output Processing

- **Final Logits Computation**
- **Softmax Activation**
- **Token Prediction**
- **Beam Search / Top-k Sampling / Temperature Scaling**

Training & Optimization

- **Objective:** Next-token prediction.
- **Loss Function:** Cross-entropy.
- **Optimizer:** AdamW with weight decay.
- **Gradient Updates:** Backpropagation.
- **Checkpointing & Regularization:** Gradient checkpointing, dropout.

Post-Training Fine-Tuning

- **RLHF:** Reinforcement Learning with Human Feedback.
- **Alignment:** Reducing biases and hallucinations.
- **Multimodal Training (GPT-4+):** Vision + Text.

LLaMA 2 & LLaMA 3 Architectures – Step-by-Step In-Depth Analysis

LLaMA (Large Language Model Meta AI) models are transformer-based architectures developed by Meta (formerly Facebook). These models are highly efficient, optimized for both inference and fine-tuning, and are designed to be competitive with OpenAI's GPT series. Below is an in-depth, step-by-step architecture analysis for both **LLaMA 2** and the upcoming **LLaMA 3** models.

📌 LLaMA 2 Architecture Overview

LLaMA 2 improves upon LLaMA 1 with better training stability, longer context handling, and optimized efficiency. It consists of three main versions: **LLaMA 2-7B, LLaMA 2-13B, and LLaMA 2-65B**.

Model Foundation: Transformer Architecture

LLaMA 2 follows a **decoder-only Transformer** architecture, similar to GPT-style models but with key optimizations.

- **Multi-Layered Transformer Blocks**: Each layer consists of:
    - **Multi-Head Attention (MHA)**
    - **Feed-Forward Networks (FFN)**
    - **Rotary Positional Embeddings (RoPE)**
    - **Layer Normalization**
    - **SwiGLU Activation**
    - **Causal Attention Masking**
- **Tokenization**:
    - LLaMA 2 uses a **Byte Pair Encoding (BPE)-like tokenizer**, trained with **SentencePiece**.
    - It has a vocabulary size of **32K** tokens.

Key Optimizations Over Standard Transformer

- **Pre-normalization**:
    - Unlike traditional transformers, layer normalization is applied **before attention and feed-forward layers** rather than after.
- **Grouped-Query Attention (GQA)**:
    - LLaMA 2 reduces memory bottlenecks by grouping queries to keys/values (useful for large models like **LLaMA 2-65B**).
- **SwiGLU Activation**:
    - Instead of ReLU or GELU, LLaMA 2 uses **SwiGLU** (Swish-Gated Linear Units), which improves training efficiency and convergence.
- **Rotary Positional Embeddings (RoPE)**:
    - Replaces absolute or learned positional embeddings.
    - Allows better **long-context handling** and enables models to **extrapolate** to longer sequences.

Training Process

- **Token Mix**:
    - LLaMA 2 is trained on a **2T token dataset** from diverse sources:
        - 50% English text (Books, Wikipedia, Web Scrapes)
        - 30% Code

        - 20% Other languages & curated datasets
- **Model Sizes & Parameter Efficiency**:
    - **LLaMA 2-7B** – 7 Billion Parameters
    - **LLaMA 2-13B** – 13 Billion Parameters
    - **LLaMA 2-65B** – 65 Billion Parameters
- **Training Hardware**:
    - Trained on **2048 A100 GPUs** (for LLaMA 2-65B).
    - Used **FSDP (Fully Sharded Data Parallel) & ZeRO optimizations**.

📌 LLaMA 3 Architecture (Upcoming)

LLaMA 3 is expected to bring major advancements in efficiency, context length, and reasoning capabilities.

Expected Architectural Enhancements

LLaMA 3 is rumored to have:

- **More Efficient Attention Mechanisms**:
    - **Mixture of Experts (MoE)** – dynamically activates only subsets of the model per token.
    - **Sliding Window Attention** for **longer context lengths** (possibly 64K tokens).
- **Enhanced Memory Management**:
    - Hybrid of **KV Cache Optimization** + **FlashAttention 2** for better inference speed.
- **Upgraded Tokenization**:
    - LLaMA 3 might increase vocabulary size (~64K tokens) for better multilingual capabilities.

Larger & More Diverse Training Data

- Meta is reportedly training **1T+ token datasets**, expanding into **code-heavy, multi-lingual, and multimodal** sources.
- **Fine-tuning capabilities**:
    - LLaMA 3 will likely support **efficient LoRA adapters** and **QLoRA** (quantization-aware fine-tuning).

Training with Next-Gen Hardware

- Expected to be trained on **H100 GPUs with FP8 precision**, reducing compute costs.
- Will leverage **DeepSpeed ZeRO-3**, **Tensor Parallelism**, and **dynamic batch scheduling**.

📌 Step-by-Step Architecture Breakdown

Component	LLaMA 2	Expected in LLaMA 3
Model Type	Decoder-only Transformer	Decoder-only Transformer (Optimized)
Positional Embeddings	RoPE	Extended RoPE/Hybrid Methods
Attention	Multi-Head Attention + GQA	Mixture of Experts (MoE) + Windowed Attention
Feedforward	SwiGLU Activation	SwiGLU + Sparse Activation
Memory Optimization	KV Cache	FlashAttention 2
Context Length	4K Tokens	32K–64K Tokens
Tokenization	SentencePiece (32K vocab)	Extended Vocab (~64K)
Training Data	2T tokens, mostly English, some code	5T+ tokens, more code & multi-modal
Inference Optimization	Quantization & LoRA Fine-tuning	MoE, QLoRA & Adaptive Compute
Compute Used	A100 GPUs (2048)	H100 GPUs with FP8 Precision

📌 Summary

- **LLaMA 2** is an efficient **decoder-only transformer** with **GQA, SwiGLU, and RoPE**.
- **LLaMA 3** is expected to **scale massively**, bringing **longer context, MoE, and hybrid attention**.
- Anthropic's Claude series comprises advanced large language models (LLMs) designed to perform a wide range of tasks with varying levels of capability, speed, and cost. The Claude 3 family, introduced in March 2024, includes three primary models: Claude 3 Haiku, Claude 3 Sonnet, and Claude 3 Opus. Each model is built upon a transformer-based architecture and incorporates unique features tailored to specific use cases.

Claude 3 Haiku

- Claude 3 Haiku is engineered for rapid responses and cost-effectiveness, making it ideal for applications requiring near-instantaneous outputs. Despite its compact size, Haiku maintains a high level of intelligence suitable for tasks such as content moderation, quick translations, and summarization of unstructured data. It supports a context window of up to 200,000 tokens, enabling it to process substantial amounts of information swiftly.

Claude 3 Sonnet

- Positioned as a balanced solution, Claude 3 Sonnet offers an optimal mix of intelligence and speed. It is well-suited for enterprise workloads that demand both performance and efficiency. Sonnet excels in tasks like knowledge retrieval, sales automation, and complex data analysis. It also features a 200,000-token context window, allowing it to handle extensive documents and datasets effectively.

Claude 3 Opus

- As the most advanced model in the Claude 3 family, Opus is designed for highly complex tasks requiring superior reasoning and comprehension. It demonstrates near-human levels of understanding and fluency, making it suitable for sophisticated dialogue, creative content generation, intricate coding challenges, and scientific inquiries. Opus also supports a 200,000-token context window, facilitating the processing of large-scale inputs such as entire codebases or comprehensive financial reports.

Architectural Features

- All models in the Claude 3 series are built upon a transformer-based architecture, which is standard in modern LLMs. This architecture enables the models to understand and generate human-like text by processing input data through layers of attention mechanisms. A notable feature of the Claude 3 models is their extensive context window of 200,000 tokens, allowing them to maintain context over long inputs and generate coherent, contextually relevant outputs.
- In addition to their transformer architecture, the Claude 3 models incorporate multimodal capabilities, enabling them to process and analyze visual data alongside textual information. This feature enhances their versatility, allowing for tasks that require understanding and interpreting images, charts, and graphs.
- Furthermore, the Claude 3 series is developed using Anthropic's Constitutional AI framework, which guides the models' behavior to align with human values and ethical considerations. This framework involves training the models with a set of principles designed to promote helpfulness, honesty, and harmlessness, ensuring that the AI's outputs are both useful and aligned with societal norms.
- In summary, the Claude 3 family offers a range of models tailored to different needs, from rapid, cost-effective responses to advanced, complex reasoning tasks. Their

transformer-based architecture, extensive context windows, multimodal capabilities, and adherence to ethical AI principles make them versatile tools for a variety of applications.

Falcon LLM Architecture (Falcon-40B & Falcon-7B)

Falcon LLM, developed by **Technology Innovation Institute (TII)**, is optimized for high performance and efficiency. It uses a **Transformer-based decoder-only** architecture.

Step-by-Step Falcon Architecture

Model Overview

- **Type**: Decoder-only Transformer
- **Size**: Falcon-7B (7B parameters), Falcon-40B (40B parameters)
- **Training Data**: Refined web data (RefinedWeb)
- **Optimization Techniques**:
  - Multi-Query Attention (MQA)
  - Flash Attention
  - LayerNorm Optimization

Key Components

1. **Token Embedding Layer**
   - Converts input tokens into dense vector representations.
   - Learned embeddings with position encodings.
2. **Multi-Query Attention (MQA)**
   - Unlike Multi-Head Attention (MHA), MQA uses multiple query heads but only **one key and value head**.
   - Improves GPU efficiency and reduces memory footprint.
3. **Rotary Positional Embeddings (RoPE)**
   - Provides a continuous representation of token positions.
   - Enables long-context learning without explicit position encodings.
4. **LayerNorm Pre-Normalization**
   - Applies LayerNorm before Attention and MLP layers (instead of after).
   - Leads to more stable training.
5. **MLP (Feedforward Block)**
   - Uses **SwiGLU activation** instead of ReLU/GELU.
   - Improves gradient flow and efficiency.
6. **Final Layer Norm & Output Head**
   - Normalizes the outputs.
   - Connects to the vocabulary projection layer.

Falcon Optimizations

- **FlashAttention**: Optimized memory-efficient attention.
- **Parallelism Techniques**:

    - **TP (Tensor Parallelism)**: Splits model across multiple GPUs.
    - **PP (Pipeline Parallelism)**: Splits forward and backward pass.
    - **FSDP (Fully Sharded Data Parallelism)**: Memory-efficient training.
- **Inference Optimizations**: **Continuous Batching** for real-time inference.

MosaicML MPT (Mosaic Pretrained Transformer)

MPT models, developed by **MosaicML**, focus on efficiency and scalability, offering:

- **MPT-7B** (base)
- **MPT-30B** (larger)
- **MPT-7B-StoryWriter** (fine-tuned for long-context)

Step-by-Step MPT Architecture

Model Overview

- **Type**: Decoder-only Transformer (similar to Falcon)
- **Size**: 7B, 30B parameters
- **Training Data**: High-quality dataset for robust generalization
- **Optimization Techniques**:
    - ALiBi (Attention with Linear Biases)
    - FlashAttention for efficiency
    - Efficient memory management with `PagedAttention`

Key Components

1. **Token Embedding Layer**
    - Similar to Falcon.
    - Converts token indices to embeddings.
2. **Attention with ALiBi (Attention with Linear Biases)**
    - Unlike RoPE, ALiBi **does not add positional embeddings explicitly**.
    - Instead, **biases the attention scores** based on distance.
    - Enables long-context learning without increasing memory usage.
3. **Multi-Head Attention with FlashAttention**
    - Optimized for large batch sizes.
    - Improves efficiency by reducing redundant memory operations.
4. **Feedforward Block (MLP)**
    - **Uses SwiGLU activation** (same as Falcon).
    - Optimized for throughput.
5. **Norm Layers and Residual Connections**
    - Pre-layer norm (before attention & MLP).
    - Essential for stable training.
6. **PagedAttention for Efficient Memory Use**

    - Designed for long-context models.
    - Handles attention computation in chunks (pages) to avoid memory overflow.

MPT Optimizations

- **Continuous Batching for Inference**: Efficient response generation.
- **PagedAttention for Long Contexts**: Optimized memory handling.
- **Layer-wise Scaling for Stability**: Improved model convergence.

Other Open-Source LLMs

LLaMA (Meta AI)

- **Type**: Decoder-only Transformer.
- **Key Optimizations**:
    - Pre-normalization (like Falcon).
    - SwiGLU activation.
    - Rotary Positional Embeddings (RoPE).
    - Smaller model variants (7B, 13B, 30B, 65B).

BLOOM (BigScience)

- **Type**: Auto-regressive Transformer.
- **Key Optimizations**:
    - GELU activation.
    - Multi-head attention.
    - Trained on 46 languages and multiple programming languages.

OpenLLaMA

- **Type**: Replication of LLaMA with open weights.
- **Key Optimizations**:
    - Trained on fully open datasets.
    - Matches Meta's LLaMA-7B performance.

GPT-J & GPT-NeoX (EleutherAI)

- **Type**: GPT-style auto-regressive Transformer.
- **Key Optimizations**:
    - Optimized for large-scale text generation.
    - Uses efficient parallelism strategies.

Comparative Analysis

Model	Type	Positional Encoding	Attention Optimization	Key Feature
Falcon	Decoder-only Transformer	RoPE	Multi-Query Attention	High efficiency & low memory usage
MPT	Decoder-only Transformer	ALiBi	FlashAttention	Long-context support
LLaMA	Decoder-only Transformer	RoPE	Multi-head Attention	Lightweight and scalable
BLOOM	Decoder-only Transformer	Learned Positional	Standard Multi-head	Multilingual capabilities
GPT-J	Decoder-only Transformer	Learned Positional	Standard Multi-head	GPT-style architecture

---

Conclusion

- **Falcon** and **MPT** are optimized for efficient memory use and long-context processing.
- **LLaMA** is a lightweight, efficient alternative to GPT-3.
- **BLOOM** excels in multilingual generation.
- **GPT-J** and **NeoX** are alternatives for general text generation.

BERT (Bidirectional Encoder Representations from Transformers)

BERT is an **encoder-only** model based on the Transformer architecture. It is **bidirectional**, meaning it considers context from both the left and right of a given token, making it highly effective for **NLP tasks requiring deep understanding**, such as **question answering and sentiment analysis**.

Architecture of BERT

- **Input Representation**
  - Input tokens are **WordPiece embeddings** with subword tokenization (e.g., "playing" → ["play", "##ing"]).
  - Each token consists of:
    - **Token embeddings** (word representations)
    - **Segment embeddings** (distinguishing sentence pairs)
    - **Positional embeddings** (position in the sequence)
- **Transformer Encoder (Bidirectional Self-Attention)**

    - Uses **stacked Transformer encoder layers (e.g., 12 in BERT-Base, 24 in BERT-Large)**.
    - Each encoder has:
        - **Multi-Head Self-Attention (MHSA)** to capture bidirectional dependencies.
        - **Feed-forward layers** (fully connected networks with GELU activation).
        - **Layer Normalization** and **Dropout**.
- **Pretraining Tasks**
    - **Masked Language Modeling (MLM)**
        - 15% of input tokens are randomly masked (`[MASK]`).
        - The model predicts the original tokens based on context.
    - **Next Sentence Prediction (NSP)**
        - The model predicts if two sentences are consecutive.
- **Fine-tuning**
    - Adapted for downstream tasks such as **question answering (SQuAD), sentiment classification, and named entity recognition**.

BERT Summary

- **Type:** Encoder-only Transformer.
- **Training Objective:** MLM + NSP.
- **Strength:** Deep contextual understanding.
- **Weakness:** Cannot generate text (only classification and span-based tasks).

GPT (Generative Pretrained Transformer)

GPT is a **decoder-only** Transformer model designed for **text generation**. Unlike BERT, it is **unidirectional**, meaning it generates text **left to right**, predicting one token at a time.

Architecture of GPT

- **Input Representation**
    - Uses **Byte Pair Encoding (BPE)** for tokenization.
    - Positional embeddings are **added** to token embeddings.
- **Transformer Decoder Stack (Causal Self-Attention)**
    - Uses **stacked Transformer decoder layers (e.g., 12 in GPT-2 Small, 96 in GPT-4 Large)**.
    - Each decoder has:
        - **Masked Multi-Head Self-Attention (Masked MHSA)**
            - Ensures the model **only sees previous tokens** to prevent peeking at future words.
        - **Feed-forward layers** (with GELU activations).
        - **Layer Normalization** and **Dropout**.
- **Pretraining Task**

- **Causal Language Modeling (CLM)**
    - The model is trained to predict the next token given previous tokens.
    - Example:
        - Input: "The sky is"
        - Target: "blue"
- **Fine-tuning & RLHF (for GPT-4, ChatGPT)**
    - Fine-tuned for **chatbots, dialogue systems, and creative text generation**.
    - Reinforcement Learning from Human Feedback (RLHF) refines responses.

GPT Summary

- **Type:** Decoder-only Transformer.
- **Training Objective:** CLM (causal language modeling).
- **Strength:** Strong text generation capabilities.
- **Weakness:** Struggles with bidirectional understanding compared to BERT.

T5 (Text-to-Text Transfer Transformer)

T5 is a **sequence-to-sequence (encoder-decoder) Transformer** that converts **every NLP task into a text-to-text problem**.

Architecture of T5

- **Input Representation**
    - Uses **SentencePiece tokenization** (based on BPE).
    - Converts **all tasks into text format**.
    - Example tasks:
        - **Summarization:** `"summarize: The article is about..."` → `"Summary: ..."`.
        - **Translation:** `"translate English to French: Hello"` → `"Bonjour"`.
- **Encoder-Decoder Transformer**
    - **Encoder:**
        - Similar to BERT, but does not use MLM.
        - Uses **full bidirectional self-attention**.
    - **Decoder:**
        - Similar to GPT, uses **causal attention**.
        - Predicts tokens autoregressively (one by one).
- **Pretraining Task: Span Corruption**
    - Instead of MLM, T5 **replaces spans of text with a single `[MASK]` token** and requires the model to generate the missing span.
    - Example:
        - Input: `"The Eiffel [MASK] is in Paris."`

        - Target: `"Tower"`
- **Fine-tuning**
    - Used for **text generation, summarization, translation, question answering, and more**.

T5 Summary

- **Type:** Encoder-Decoder Transformer.
- **Training Objective:** Span corruption.
- **Strength:** Flexible for **both understanding and text generation**.
- **Weakness:** More complex and computationally expensive than BERT/GPT.

Comparison Table

Model	Architecture	Attention Mechanism	Training Objective	Strength	Weakness
BERT	Encoder-only	Bidirectional Self-Attention	MLM + NSP	Strong contextual understanding	Cannot generate text
GPT	Decoder-only	Causal Self-Attention	CLM (predict next token)	Strong text generation	Weak bidirectional understanding
T5	Encoder-Decoder	Full (Encoder) + Causal (Decoder)	Span corruption	Versatile (both understanding & generation)	Expensive computation

---

When to Use Which Model?

- **Use BERT** if you need **text classification, named entity recognition, or QA (without generation)**.
- **Use GPT** if you need **chatbots, creative text generation, or auto-completion**.
- **Use T5** if you need **both understanding and text generation, such as summarization or translation**.

Final Thoughts

Each model has a unique architecture that makes it suitable for different tasks:

- **BERT** (Encoder) → Best for **understanding**.

- **GPT** (Decoder) → Best for **generation**.
- **T5** (Encoder-Decoder) → Best for **combining both**.

Standard Metrics for GenAI Tasks

While user experience is essential, there are well-established metrics that provide **objective evaluation** for various GenAI tasks:

Summarization (e.g., T5, BART, Pegasus)

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**
  - **ROUGE-1, ROUGE-2**: Measures overlap of unigrams and bigrams.
  - **ROUGE-L**: Measures longest common subsequence.
  - **ROUGE-S**: Measures skip-grams.
- **BLEU (Bilingual Evaluation Understudy):** Originally for translation, but sometimes used for summarization.
- **METEOR**: Considers synonyms, stemming, and order, making it more sophisticated than BLEU.
- **BERTScore:** Uses contextual embeddings from BERT to compute similarity.

Question Answering (e.g., GPT, Llama, FLAN-T5)

- **Exact Match (EM)**: Checks if the model's answer exactly matches the ground truth.
- **F1-score**: Evaluates overlap between predicted and ground truth answer.
- **BLEU / METEOR**: For longer answer generations.
- **BERTScore**: Measures contextual similarity of answers.
- **G-Eval**: A human-aligned metric based on GPT evaluations.

Text Generation (e.g., GPT, LLaMA, Falcon)

- **Perplexity (PPL)**: Measures how well a model predicts a sample (lower is better).
- **Diversity Metrics**:
  - **Self-BLEU**: Lower values mean more diverse generations.
  - **Distinct-n**: Measures the uniqueness of n-grams (higher is better).
- **BERTScore / BLEU / METEOR**: For reference-based evaluation.
- **Toxicity Score** (Perspective API, OpenAI Moderation API): Checks for safety.

Translation (e.g., NLLB, M2M-100, MarianMT)

- **BLEU**
- **METEOR**
- **CHRF++ (Character F-score)**: Works better for morphologically rich languages.
- **BERTScore**: Captures contextual meaning.

Code Generation (e.g., CodeLlama, StarCoder, GPT-4 Code)

- **Code BLEU**: BLEU + exact match + data-flow consistency.
- **Pass@k**: Measures how often the correct code appears in the top K generations.
- **Execution Accuracy**: Runs the generated code to check correctness.

Image-to-Text (e.g., BLIP, Flamingo)

- **CIDEr (Consensus-based Image Description Evaluation)**: Measures relevance.
- **SPICE (Semantic Propositional Image Caption Evaluation)**: Focuses on objects and relations.
- **BLEU / METEOR / ROUGE**: Also used for evaluating text descriptions.

Human Evaluation & User Experience

Besides metrics, human evaluation remains **critical** for assessing:

- **Coherence**: Logical flow of responses.
- **Fluency**: Grammar and readability.
- **Relevance**: Whether the response is on-topic.
- **Faithfulness**: Absence of hallucinations.
- **Bias & Fairness**: Ensuring ethical AI behavior.

Automated vs. User-Based Evaluation

- **Automated Metrics**: Good for quick benchmarking and large-scale comparisons.
- **Human Evaluation**: Needed for subjective tasks like creativity, humor, or reasoning.