Overview: RAG, LLMs, SAM, and GANs

# 1. Retrieval-Augmented Generation (RAG)

## What is RAG?

Retrieval-Augmented Generation (RAG) is an architecture that combines a pre-trained language model (like BERT or GPT) with an external knowledge retriever to enhance its output by grounding it in relevant external documents. RAG works in two main steps: first, it encodes a user query and uses vector similarity search to retrieve the most relevant documents from a large corpus (via dense retrievers like FAISS); then, it feeds these retrieved chunks along with the original query into a generator (like a seq2seq model or decoder-only LLM) to produce a final, more factually grounded response. This framework helps mitigate hallucinations and improves factual accuracy.

## RAG Architecture

The architecture consists of two primary components: (1) a retriever that embeds the query and searches an indexed document corpus, and (2) a generator that synthesizes the final answer using the query and top-k retrieved documents. The retriever is typically based on dense embedding models like DPR (Dense Passage Retrieval), and the generator is a transformer-based model like BART or GPT. This fusion enables LLMs to answer domain-specific or long-tail questions without needing to be retrained.

# 2. Large Language Models (LLMs)

## How LLMs Work

LLMs are deep learning models built primarily using the Transformer architecture, capable of understanding and generating human-like text at scale. These models are trained on vast amounts of text data using self-supervised learning, where the model learns to predict masked tokens (in encoder models) or the next word (in decoder-only models). Their architecture typically comprises multiple layers of attention mechanisms and feed-forward neural networks.

## Decoder-Only Transformers

A prominent type of LLM is the decoder-only Transformer, which processes input tokens sequentially and generates output tokens autoregressively. This means the model generates one token at a time while attending only to past tokens. GPT (Generative Pre-trained Transformer) is the best-known decoder-only LLM. It consists of stacked transformer decoder blocks, each containing masked self-attention layers and position-wise feedforward layers. Its ability to scale up with more parameters and data has led to state-of-the-art performance in text generation and reasoning tasks.

# 3. Vision Transformers (ViT) and SAM

## What is ViT?

Vision Transformers (ViTs) apply the transformer architecture to image data. Instead of using convolutional layers like CNNs, ViTs split an image into patches, flatten them, and treat each patch as a token in a sequence. These tokenized patches are then fed into standard transformer encoder layers with self-attention. ViTs are powerful for large-scale vision tasks such as image classification, segmentation, and object detection.

## SAM (Segment Anything Model)

A specialized application of ViT is the Segment Anything Model (SAM) by Meta AI. SAM combines a promptable ViT encoder with lightweight prompt encoders (points, boxes, masks) and a mask decoder. The encoder generates image embeddings, and prompts guide the decoder to predict segmentation masks for arbitrary objects. SAM supports zero-shot generalization and enables flexible, fast segmentation, making it suitable for many downstream vision tasks.

# 4. Generative Adversarial Networks (GANs)

## GAN Architecture

GANs are a class of generative models that consist of two neural networks: a Generator and a Discriminator, trained in opposition. The Generator learns to create fake data (e.g., images) that mimics real data, while the Discriminator learns to distinguish between real and generated data. The Generator's goal is to fool the Discriminator, and the Discriminator's goal is to correctly classify real vs. fake samples.

## How GANs Work

The architecture of a GAN includes:  
Generator (G): Takes random noise as input and outputs synthetic data. Typically composed of dense and transpose convolutional layers.  
Discriminator (D): Takes data (real or fake) as input and outputs a probability of authenticity. Uses convolutional layers to process input.  
  
Training involves a minimax game where G tries to minimize the Discriminator's ability to detect fakes, and D tries to maximize it. Over time, the Generator improves to the point where it produces highly realistic outputs. Variants like DCGAN, WGAN, CycleGAN, and StyleGAN improve training stability and output quality.