Key Models

GPT (Generative Pre-trained Transformer)

GPT is a family of language models developed by OpenAI. These models are trained to generate human-like text and can perform a wide range of natural language processing (NLP) tasks, such as answering questions, writing essays, creating dialogue, summarizing, translating, and more all based on the input it receives.

Core Idea:

GPT uses a Transformer-based architecture, which allows it to understand context in sequences (such as a paragraph or conversation). It works autoregressively, meaning it generates text one token at a time by predicting the next word based on all the previous words.

Architecture Details:

- Model Type: Transformer Decoder (only the decoder half of the transformer architecture is used)
- Training Objective: Predict the next word (token) in a sequence this is called causal language modeling.
- Input/Output: Given a prompt like "The sun is", it might output "shining brightly over the ocean".

Training Process:

1. **Pretraining**: The model is trained on a massive dataset scraped from the internet (books, Wikipedia, articles, web pages, code).
2. **Unsupervised learning**: The model doesn't need labeled data; it learns by predicting missing parts of text.
3. **Fine-tuning (optional)**: Later versions like GPT-3.5 or GPT-4 may go through fine-tuning or reinforcement learning from human feedback (RLHF) to align the model with user expectations and reduce harmful outputs.

Major Versions:

- GPT-1 (2018): Proof of concept (117M parameters)
- GPT-2 (2019): More powerful, but initially not released due to concerns about misuse (1.5B parameters)
- GPT-3 (2020): Breakthrough model with 175B parameters
- GPT-3.5 / ChatGPT: Refined version used in free-tier ChatGPT
- GPT-4 (2023): Even more capable, supports multimodal inputs (text + images)
- GPT-4 Turbo: Optimized for efficiency, used in paid ChatGPT services

Capabilities:

- Text generation, summarization, translation
- Reasoning and problem-solving (to a limited extent)
- Creative writing (stories, poems, scripts)
- Answering factual questions
- Assisting in tasks like brainstorming or coding

Limitations:

- Hallucination: Can generate incorrect or fictional facts
- Bias: Reflects biases in training data
- Context window limit: Can "forget" earlier parts of a long conversation (though newer versions have larger context windows)

DALL·E

DALL·E is a text-to-image generation model developed by OpenAI. It can generate high-quality images from natural language descriptions, such as *“a futuristic city on Mars”* or *“a cat in a space suit sitting on a beach.”*

The name is a combination of:

- **"Dali"** – referencing surrealist artist Salvador Dalí
- **"WALL·E"** – referencing the Pixar robot, representing the AI part

Core Idea:

DALL·E takes a text prompt and converts it into an image that visually represents the description. It “understands” both linguistic and visual concepts, enabling it to produce imaginative or realistic scenes from scratch.

How It Works (DALL·E 2 and 3):

DALL·E 2:

- Combines CLIP (Contrastive Language–Image Pretraining) and diffusion models
- Uses prior models to convert text into an image embedding
- A decoder then generates an image that matches that embedding

DALL·E 3:

- Built into GPT-4 for tighter integration with language understanding
- Handles complex prompts much better, with improved coherence and text rendering (e.g., in signs, logos)
- More creative and controllable compared to DALL·E 2

Training Process:

1. Trained on text-image pairs (e.g., images with captions or descriptions)
2. Learns the relationship between words and visual concepts
3. Can also do inpainting (filling in missing parts) and variation (create multiple images with the same concept)

Use Cases:

- Generating concept art or illustrations from text
- Creating content for games, ads, or storytelling

- Designing scenes or compositions for films or books

Examples:

- DALL·E 2 / 3 (OpenAI)
- Integrated into ChatGPT (Pro version) as of GPT-4 Turbo

Limitations:

- May struggle with very abstract or contradictory prompts
- Earlier versions had trouble rendering text in images
- Potential for misuse (e.g., deepfakes, misinformation)

Codex

Codex is a language model specifically trained to generate and understand computer code. It is a descendant of GPT-3 but fine-tuned on billions of lines of publicly available code (from GitHub and other sources).

It powers tools like GitHub Copilot, enabling AI-assisted programming directly in development environments.

Core Idea:

Codex takes natural language instructions and converts them into code in various programming languages (Python, JavaScript, HTML/CSS, etc.). It understands both the human intent (from English) and how to express that intent in code.

Architecture:

- Based on the GPT family (GPT-3, specifically)
- Fine-tuned on code data: functions, libraries, documentation
- Supports dozens of programming languages

How It Works:

1. Given a prompt like:

"Write a Python function that returns the factorial of a number"

2. Codex uses its trained understanding of programming syntax and logic to generate the correct function.
3. It can also:
    - Translate code between languages (e.g., Java → Python)
    - Add comments or documentation
    - Fix bugs or improve performance
    - Autocomplete code in real-time as you type

Use Cases:

- Writing boilerplate code quickly
- Assisting junior developers in learning
- Speeding up debugging or code documentation
- Generating simple web apps or prototypes

Examples:

- GitHub Copilot (from GitHub + OpenAI)
- Replit AI
- CodeWhisperer (by AWS) (Codex-inspired)

Limitations:

- May generate insecure or incorrect code
- Can suggest deprecated or non-optimal patterns
- Not a substitute for understanding developers still need to review and validate code

Stable Diffusion

Stable Diffusion is an open-source text-to-image diffusion model developed by Stability AI, in collaboration with CompVis and LAION. It creates high-quality, photorealistic or stylized images from textual prompts.

Unlike proprietary models like DALL·E or Midjourney, Stable Diffusion is freely available, making it a major milestone in democratizing generative image AI.

Core Idea:

Stable Diffusion uses latent diffusion instead of generating full-size images pixel by pixel, it first compresses images into a latent space, does the generation there, and then decodes the result back into an image.

This drastically reduces compute requirements while preserving image quality.

How It Works:

1. Text Prompt is processed using a language encoder (like CLIP or T5) to understand the meaning.
2. A diffusion process begins with random noise in the latent space.
3. The model gradually denoises this latent representation until it resembles an image that matches the prompt.
4. A decoder (VAE) converts the latent image back into pixel space.

Advantages of Latent Diffusion:

- Faster and more memory-efficient
- Enables real-time generation on consumer-grade GPUs (e.g., 6GB VRAM)
- Open-source → customizable and extendable by developers

Use Cases:

- Creating AI art for social media, books, or games
- Style transfer (applying one image's style to another)
- Generating avatars, concept designs, or wallpapers

Examples:

- Stable Diffusion 1.5 – Earlier release, widely adopted
- Stable Diffusion XL (SDXL) – Improved text understanding and detail

Limitations:

- May require prompt engineering to get precise results
- Can sometimes produce anatomical errors (especially in humans/hands)
- Needs post-processing or upscaling for commercial quality in some cases

Diffusion Models

Diffusion models are a class of generative models designed to create data (images, audio, etc.) by learning how to reverse a gradual noising process. Instead of generating samples directly, they start from pure noise and iteratively refine it into meaningful data.

How Diffusion Models Work (Step-by-Step):

1. **Forward Process (Adding Noise):**
   During training, clean data (like an image) is progressively corrupted by adding Gaussian noise over many steps until it becomes almost pure noise.
2. **Reverse Process (Denoising):**
   The model learns to reverse this process: starting from noise, it predicts how to remove the noise step-by-step to reconstruct the original data.
3. **Training Objective:**
   The model is trained to predict the added noise at each step, effectively learning a mapping from noisy data back to clean data.
4. **Generation:**
   At inference, the model starts with random noise and applies this

learned denoising process repeatedly, producing a clean, high-fidelity sample.

Strengths:

- Stable training: Unlike GANs, diffusion models don't require adversarial training, reducing instability and mode collapse.
- High-quality outputs: Generate sharp, diverse, and detailed images.
- Flexibility: Applicable to images, audio, and other continuous data types.

Examples & Applications:

- Stable Diffusion: Efficient, open-source text-to-image model using latent diffusion.
- DALL·E 2/3: OpenAI’s text-to-image system based on diffusion.
- Audio synthesis: Can be adapted for generating raw audio waveforms.

GANs (Generative Adversarial Networks)

GANs are generative models introduced in 2014 by Ian Goodfellow, which use two neural networks competing against each other to produce realistic data.

How GANs Work (Step-by-Step):

1. **Generator:**
   Takes random noise (latent vector) and generates fake data (e.g., images).
2. **Discriminator:**
   Receives both real data and generated data and learns to distinguish between them.
3. **Adversarial Training:**

    - The generator tries to fool the discriminator by producing increasingly realistic data.
    - The discriminator improves its ability to detect fakes.
    - This adversarial game continues until the generator produces data indistinguishable from real data.

Strengths:

- Capable of producing very sharp and realistic images.
- Well-suited for image-to-image translation, style transfer, super-resolution, and inpainting.
- Can learn complex distributions with fewer training samples compared to some other methods.

Challenges:

- Training instability: GANs can be hard to train due to the delicate balance needed between generator and discriminator.
- Mode collapse: The generator may produce limited diversity by focusing on certain patterns.
- Sensitive to hyperparameters and architecture choices.

Variants of GANs:

- StyleGAN: Famous for generating high-res, photorealistic faces with fine control over style.
- CycleGAN: Used for unpaired image-to-image translation (e.g., horses to zebras).
- Pix2Pix**:** For paired image translation tasks.

Transformers

Transformers are deep learning architectures that revolutionized natural language processing and have been adapted for other data types like images and audio. They excel at handling sequences by capturing long-range dependencies.

How Transformers Work:

1. **Self-Attention Mechanism:**
   Instead of processing sequences step-by-step (like RNNs), transformers use self-attention to weigh the importance of every token in the sequence relative to every other token simultaneously. This allows the model to capture context across long distances efficiently.
2. **Encoder-Decoder Architecture:**
   - The **encoder** processes the input sequence to create contextualized representations.
   - The **decoder** generates output sequences, attending to encoder outputs and previously generated tokens (in tasks like translation).
3. **Transformer Decoder-Only Models:**
   Models like GPT use only the decoder stack to perform autoregressive generation predicting the next token given prior tokens.

Key Components:

- Multi-head Attention: Multiple attention “heads” run in parallel to capture different types of relationships.
- Positional Encoding: Since transformers process tokens in parallel, positional encodings inject information about token order.
- Feed-forward Networks: Applied after attention to each token independently, adding non-linearity.

Strengths:

- Parallel processing: Enables training on massive datasets efficiently.

- Scalability: Performance improves with model size and data.
- Versatility: Successfully applied beyond text image generation, audio, and code generation.
- Context awareness: Can consider long context windows (thousands of tokens in latest versions).

Examples:

- GPT series (OpenAI): Autoregressive text generation.
- BERT (Google): Encoder-only model for understanding tasks.
- Vision Transformers (ViT): Adapt transformer architecture for image classification.
- Multimodal transformers: Combine text and images, e.g., CLIP or Flamingo.