**GPT , Dall-E, Codex, Stable-Diffusion**

**GPT**

GPT stands for generative pre-trained transformer and is a family of neural network models that analyze data and interpret and produce human-like text, images, and sounds. People and organizations use GPT to summarize long text and meetings, translate languages, create written communication, write code, generate images, and answer questions in a conversational tone.

**What GPT is and how it works**

GPT is a [deep learning](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-deep-learning/) neural network that analyzes prompts made up of natural language, images, or sounds to predict the best possible response based on its interpretation of the input. To do this, it’s trained with massive datasets using hundreds of billions of parameters. GPT references that learning to weight the importance of different components in a sequence, such as words in a sentence or parts of images or sounds. The weighting allows it to infer relevance and context so that it can generate content that makes sense with the prompt.

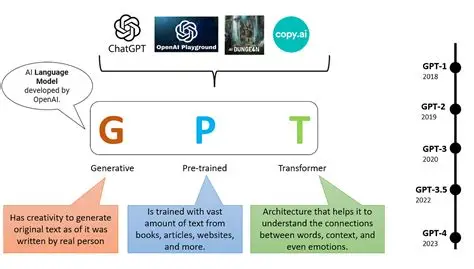


**Training overview**

To be effective, GPT must be able to parse and interpret a myriad of prompts and requests. It prepares for this by training on massive datasets, including large text corpora, using unsupervised deep learning, a subset of [machine learning](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-machine-learning-platform/). In unsupervised learning the model teaches itself to find patterns in unlabelled data without guidance from humans. GPT uses [computer vision](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision/#object-classification) to identify and understand objects and people in images.  
  
GPT can also be trained for very specific scenarios, such as for an industry, like banking or law. In these instances, supervised learning is used, which means that training data is labeled by humans.

**Basic GPT architecture**

GPT is built on the transformer architecture, which uses the self-attention mechanism to analyze different components of a prompt and their relationship to each other to interpret context and meaning. For example, the word “cloud” can refer to condensed vapor in the sky or, as in [cloud computing](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing/), a technology platform. People and GPT determine which version of the word is appropriate by evaluating the meaning of the other words surrounding it in a sentence or paragraph.  
  
The transformer architecture is able to do this by turning words and their meaning into mathematics. It breaks up text, images, and sounds into smaller pieces called tokens. The tokens are assigned a vector, which encodes meaning. The encoded vectors, called embeddings, are then sent through an attention block where they exchange information and make updates to the vectors as appropriate. Once GPT has determined the meaning of the prompt, it produces a prediction in the form of a probability distribution and suggests the next word, image, or sound in the sequence. By repeating this process over and over, it can write long passages or carry on a conversation.



**Key components**

**The architecture is made up of two parts:**

* **Encoder.** The encoder is the part of the system that breaks down text, images, and sounds into mathematical embeddings. Each embedding is assigned a weight, which tells it how relevant it is to the context and meaning. The embeddings are then compared to each other using the self-attention mechanism to further refine their meaning.
* **Decoder.** The decoder uses the vectors and weights to determine possible outputs and predict the best one. Because the most current versions of GPT have been trained on so much data, they’ve gotten quite good at using this process to write fluent and coherent text.

The benefits and challenges of GPT

GPT has the potential to transform how you and your organization work, helping you save time and money. But there are also risks with using this technology without careful guardrails. It’s critical to always carefully vet the information you get from GPT or any other AI system to confirm it’s accurate and ethical.

Benefits

* **Simplify research.** GPT can scour the internet and/or other data sources and provide a summary of what it found and sources if requested.
* **Enhance computer code.** Developers use GPT to help them write new code or simplify what they’ve already written.
* **Write faster.** One of the most popular ways to use GPT is as a writing tool. It can quickly synthesize a lot of information and develop reports, blog posts, emails, and other written materials.
* **Reduce busywork.** GPT can do things like summarize meetings, translate languages, and answer questions, empowering you to spend more time on more impactful tasks.
* **Boost creativity.** In addition to writing poetry, GPT can quickly generate lots of different ideas, making it a great tool for brainstorming.
* **Customize to your business.**GPT can be trained to meet the unique needs of different organizations and industries.

Challenges

* **Bias.**Like all AI models that rely on human-created data, the biases inherent in that data may make it into the GPT output. For example, AI models may assume that certain roles in society, like scientist, are only performed by men because most of the historical data is about male scientists.
* **Inaccuracies.** Because GPT generates output based on a prediction, it isn’t always correct. Asking it to reference known materials or training it on your organization’s knowledge base can help, but a human should always review the work for accuracy.
* **Cybersecurity.** Bad actors are using GPT and other AI models to create convincing phishing emails, develop malware, and analyze organizations for vulnerabilities. Training employees to recognize phishing emails can help lower your organization’s risk. It’s also important to implement cybersecurity solutions that can detect anomalies and block malware.
* **Intellectual property violations.** The output from GPT may include images or copy created by another person or organization. Before publishing anything created by AI, confirm your organization has rights to the content and use citations appropriately.
* **Ineffective prompts.** Getting a good output from GPT requires a well-structured prompt. It may take training and trial and error to develop a prompt that gets you the results you’re hoping for.
* **Impenetrability.** Because GPT is built using a deep learning model, it’s difficult to know how it comes up with its responses, which is another reason to review its output carefully before using it.

**Use cases**

Common GPT use cases

GPT models can perform a broad range of tasks, and organizations continue to find new ways to use them in their organizations. Here are a few things to try:  
  
**Content creation.**Use GPT to help you write copy, generate memes, and produce images.  
  
**Chatbots and conversational agents.** Because GPT can understand and respond in natural language, it’s a great tool for chatbots.   
  
**Language translation.** GPT does a good job translating languages, although it’s always best to confirm accuracy with a native speaker before posting it on your website or other public space.  
  
**Sentiment analysis.** GPT can help you analyze customer reviews, social media posts, or other text to understand how people feel about your brand, products, and services.  
  
**Recommendations.** Before a big trip, consider asking GPT to recommend restaurants, hotels, and attractions to visit. With the right parameters it can help you develop a list of good options.  
  
**Research.** Because GPT is good at summarizing information, it’s also a great research tool. It can help reduce the number of websites, reports, and other documents that you need to review to find what you’re looking for. Just be sure to ask for sources so you can validate the information you get.  
  
**Meeting and document summarization.** GPT can save lots of time by providing summaries of meetings or long documents.  
  
**Code creation.**GPT knows many computer languages and can generate relevant snippets of code or explain, in conversational language, what the code is doing.  
  
**Data analysis.** Uncover trends and key insights in large datasets with the help of GPT.

**Dall-E**

DALL-E is a [**generative AI**](https://www.tutorialspoint.com/gen-ai/index.htm) tool developed by OpenAI. Its functionality is to generate images based on textual descriptions provided by the users. The model works in combination with [**natural language processing**](https://www.tutorialspoint.com/natural_language_processing/index.htm) (NLP) to interpret the prompt and computer vision to generate images.

Example of an image generated using DALL-E –

A cartoon mouse jumping into a ship

AI-generated content may be incorrect.

**Text Prompt** − A cartoon mouse wearing a sailor outfit and jumping off a cruise ship into the middle of the sea.

**Features of DALL-E**

DALL-E is developed with several advanced features to enhance its ability to generate and manipulate images from textual descriptions. Few of the features are −

Ability to Combine Multiple Objects and Their Attributes

DALL-E has the ability to understand and combine multiple objects and their attributes. For example, consider the prompt **"A red apple on a brown table with white cloth on top and grey background."** DALL-E interprets this sentence and forms associations like **(apple,red); (table,brown); (cloth,white); and (background,grey).**

A logo with a black background

AI-generated content may be incorrect.

**Enhanced Visualization Abilities**

DALL-E is developed with advanced visualization capabilities that allow users to generate images from various angles, such as zoomed-in or zoomed-out versions, internal and external displays. Along with this, the model produces realistic images by focusing on the casting of the shadow based on the orientation of the object.

**Knowledge of Geography and History**

DALL-E allows users to generate images from historic ages or an image that reflects the culture of a particular area or time period. For example, consider the prompt "Traditional Food of China." It generates an image of authentic Chinese food.

**Benefits of using DALL-E**

DALL-E is the most opted-for tool for image creation, some key benefits are −

* **Enhanced Creativity** − DALL-E allows the creation of highly creative and imaginative images that may not exist in the real world based on text description.
* **Versatility** − DALL-E can generate images from realistic portraits to fantasy landscapes, allowing diverse applications across various industries like marketing, entertainment, and education.
* **Image Quality and Customization** − DALL-E allows users to create high-quality customized images based on their needs. By giving detailed text prompts, users can generate images close to their vision.
* **Accessibility** − DALL-E generates high quality images accessible to a broader audience, including those who may not know advanced graphics or artistic skills. This tool allows users to visually express their thoughts with a simple text description.

**Limitations of using DALL-E**

While DALL-E is the most used for image generation, it has several limitations −

* **Lack of Textual Understanding** − DALL-E generates images based on text prompts, it might not fully understand the context, especially if the prompt has many attributes. This can lead to images that do not accurately represent the user's vision.
* **Ethical and Copyright Concerns** − Using DALL-E to generate images that resemble copyrighted works or mimic the style of specific artists leads to legal and ethical dilemmas.
* **Security and Misuse Risks** − Some potential risks associated with generating images using DALL-E are misuse, misleading, or harmful content.

**Future of DALL-E**

The development of DALL-E opens up a broader perspective on generative AI taking over the world and bringing revolutionary changes in various domains. Some potential directions and development for DALL-E in the future −

* Improved image quality and detail
* Better analysis of context and prompts
* Integration with other tools and platforms
* Ethical considerations and safety measures
* Enhancing customization and personalization

**Codex**

Codex has emerged as a transformative AI agent designed to augment software engineering workflows by autonomously handling tasks such as writing code, debugging, running tests, and generating pull requests. It operates as a cloud-based agent powered by codex‑1, a specialized adaptation of OpenAI’s o3 reasoning model fine‑tuned for programming contexts. Available initially to ChatGPT Pro, Team, and Enterprise users, Codex integrates directly into the ChatGPT interface, allowing developers to assign discrete tasks that run in sandboxed environments preloaded with their codebases.

**What is Codex?**

****

**Origins and Evolution**

Codex is the latest AI-driven software engineering agent developed by OpenAI, officially unveiled on May 16, 2025, as a research preview. Unlike its predecessor, the GPT series—primarily optimized for natural language tasks—Codex is rooted in a specialized derivative of the o3 model, named **codex-1**, which has been fine-tuned specifically for programming workflows . Its lineage traces back to OpenAI’s work on GPT-3 and the earlier Codex model that powers tools like GitHub Copilot, but codex-1 represents a significant leap in agentic capabilities, enabling parallel task execution and autonomous interactions with development environments.

**Core Architecture**

At its core, Codex operates as a multi-agent system hosted in the cloud. Each coding task—be it writing new features, debugging, testing, or even proposing pull requests—is dispatched to its own isolated sandbox environment preloaded with the user’s repository. This sandboxing ensures that changes are contained and reproducible, and that Codex can iteratively run tests, linters, and type checkers until tasks pass validation. The underlying **codex-1** model leverages reinforcement learning from real-world coding tasks, aligning its output closely with human coding styles and best practices.

**Purpose and Positioning**

OpenAI positions Codex as a transformative tool for software engineering teams, aiming to shift developers’ focus from routine implementation to higher-order design and orchestration work. By automating repetitive and well-specified tasks, Codex aspires to boost productivity, reduce context-switching, and embed itself within existing CI/CD pipelines. With competitors like Google’s Gemini, Anthropic’s Claude, and emerging startups in the agentic AI space, Codex serves as OpenAI’s strategic response to maintain leadership in AI-driven developer tooling .

**How does Codex work?**

**Model Architecture and Training**

Codex is powered by **codex-1**, a variant of the o3 reasoning model optimized for software engineering. Training involved two phases: a broad pretraining on large code and text corpora, followed by reinforcement learning on real-world developer tasks to refine its ability to adhere to instructions, follow repository-specific conventions, and generate test-passing code. The final model demonstrates higher accuracy in code generation, an improved understanding of repository context, and the ability to self-correct through iterative testing loops.

**Parallel Task Processing**

One of Codex’s standout features is its agentic, parallel task execution capability. Unlike single-threaded code generation tools, Codex can handle multiple concurrent assignments within a project. Each task is encapsulated in its own Docker-like sandbox, allowing developers to queue several tasks—such as implementing features, generating documentation snippets, or refactoring modules—and receive results independently, often within one to thirty minutes depending on complexity and compute availability .

**Sandboxed Execution Environment**

Security and reproducibility are paramount. Codex’s sandbox environment simulates the developer’s local setup, preloading repositories, dependencies, and configuration files. Within this isolated context, Codex can run build commands, execute test suites, invoke linters, and even interface with package managers. Upon task completion, it returns code changes, detailed test logs, and invocation results, ensuring that developers have full visibility into what was modified and why.

**Integration with ChatGPT and CLI**

For accessibility, Codex is integrated directly into the ChatGPT interface for Pro, Team, and Enterprise subscribers. Users can invoke Codex via the ChatGPT sidebar by typing natural language prompts—“Write a function to parse JSON logs” or “Fix the failing user-authentication test”—and choosing between “Code” and “Ask” modes. Additionally, Codex offers a command-line interface (CLI) that supports scripting and automation in local development environments, enabling seamless incorporation into existing workflows and CI/CD pipelines.

**How to use Codex?**

**Access and Availability**

Codex is currently available in research preview to ChatGPT Pro, Team, and Enterprise users, with an anticipated rollout to Plus and EDU users in the coming months. Access requires an active subscription ($200/month for Pro) and enrollment in the Codex preview program via the OpenAI dashboard. Users receive quota allocations based on subscription tier, reflecting the computational intensity of running codex-1. As OpenAI scales its infrastructure, availability and rate limits are expected to expand.

**Getting Started: Creating Tasks**

1. **Select Repository:** Within the ChatGPT interface, navigate to the Codex sidebar and choose the repository (either from GitHub or an uploaded ZIP).
2. **Define a Task:** Enter a natural language prompt describing the desired change or query. Prefix tasks with clear action verbs—“Implement,” “Refactor,” “Test,” or “Explain.”
3. **Choose Mode:** Click **Code** to modify code or **Ask** to query documentation or repository insights.
4. **Execute:** Codex allocates a sandbox and begins processing. A status indicator shows progress, and upon completion, you receive diffs, logs, and an execution summary.
5. **Review and Merge:** Examine suggested changes, run additional local tests if needed, and merge via your usual pull-request workflow.

**Best Practices and Tips**

* **Granular Prompts:** Smaller, well-scoped tasks yield more accurate results than broad, multi-step requests.
* **Contextual Clarity:** Provide context on coding standards, preferred libraries, and test frameworks to align Codex output with team conventions.
* **Iterative Refinement:** Use follow-up prompts to refine incomplete or suboptimal suggestions—Codex retains context within a session.
* **Sandbox Inspection:** Review sandbox logs to diagnose failures or unexpected behavior before accepting changes.

**Limitations and Considerations**

While powerful, Codex is not infallible. It may generate non-optimal code for highly specialized frameworks, mishandle edge cases, or produce inefficiencies. Network-restricted sandboxes cannot access external APIs, limiting tasks that depend on live data fetches. Moreover, computational costs and queue times can vary based on peak demand. Organizations should treat Codex outputs as suggestions, applying rigorous code review and testing before deployment.

**What are the real-world applications?**

**Feature Development**

Codex accelerates feature development by scaffolding routine components—data models, API endpoints, and UI templates. Developers can focus on core business logic while Codex generates boilerplate code and enforces project conventions automatically.

**Bug Fixing and Testing**

Automated bug triage and patch generation are among Codex’s most lauded capabilities. By supplying failing test cases or error logs, developers can prompt Codex to identify culprits, propose fixes, and validate them through sandboxed test runs, significantly reducing debugging cycles .

**Code Review and Refactoring**

Codex can perform global refactoring tasks—renaming variables, modularizing monolithic functions, or applying security patches across the codebase. It can also draft detailed pull-request descriptions, highlighting changes and rationale, which accelerates code review throughput .

**Non-Traditional Uses**

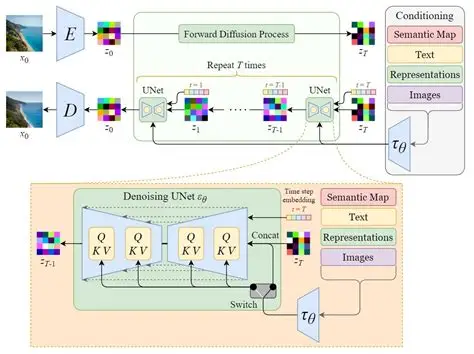
Beyond pure software engineering, Codex’s ability to interact with external services has unlocked creative applications, such as automating web form submissions, integrating with ticketing platforms to file issues, or even orchestrating simple workflows like ordering takeout via online APIs—all driven by natural language prompts .

**Conclusion**

Through its agentic design, sandboxed execution, and deep integration with ChatGPT, Codex represents a pivotal advancement in AI-driven software engineering. While still in its research preview phase, it has already begun reshaping how developers approach everyday tasks—streamlining workflows, reducing manual toil, and opening new avenues for productivity and innovation. As Codex evolves and matures, its influence on the software development lifecycle is likely to grow, heralding a new era where AI agents become indispensable partners in building the digital world.

**Stable-Diffusion**

**Stable Diffusion** is a [deep learning](https://en.wikipedia.org/wiki/Deep_learning), [text-to-image model](https://en.wikipedia.org/wiki/Text-to-image_model) released in 2022 based on [diffusion](https://en.wikipedia.org/wiki/Diffusion_model) techniques. The [generative artificial intelligence](https://en.wikipedia.org/wiki/Generative_artificial_intelligence) technology is the premier product of [Stability AI](https://en.wikipedia.org/wiki/Stability_AI) and is considered to be a part of the ongoing [artificial intelligence boom](https://en.wikipedia.org/wiki/AI_boom).



It is primarily used to generate detailed images conditioned on text descriptions, though it can also be applied to other tasks such as [inpainting](https://en.wikipedia.org/wiki/Inpainting), outpointing, and generating image-to-image translations guided by a [text prompt](https://en.wikipedia.org/wiki/Prompt_engineering).[[3]](https://en.wikipedia.org/wiki/Stable_Diffusion#cite_note-:0-3) Its development involved researchers from the Comp Vis Group at [Ludwig Maximilian University of Munich](https://en.wikipedia.org/wiki/Ludwig_Maximilian_University_of_Munich) and [Runway](https://en.wikipedia.org/wiki/Runway_(company)) with a computational donation from Stability and training data from non-profit organizations.

Stable Diffusion is a [latent diffusion model](https://en.wikipedia.org/wiki/Latent_diffusion_model), a kind of deep generative artificial [neural network](https://en.wikipedia.org/wiki/Neural_network). Its code and model weights have been released [publicly](https://en.wikipedia.org/wiki/Source-available_software),[[8]](https://en.wikipedia.org/wiki/Stable_Diffusion#cite_note-stable-diffusion-github-8) and an optimized version can run on most consumer hardware equipped with a modest [GPU](https://en.wikipedia.org/wiki/Graphics_processing_unit) with as little as 2.4 GB [VRAM](https://en.wikipedia.org/wiki/Video_random_access_memory).[[9]](https://en.wikipedia.org/wiki/Stable_Diffusion#cite_note-optimized-stable-diffusion-9) This marked a departure from previous proprietary text-to-image models such as [DALL-E](https://en.wikipedia.org/wiki/DALL-E) and [Midjourney](https://en.wikipedia.org/wiki/Midjourney) which were accessible only via [cloud services](https://en.wikipedia.org/wiki/Cloud_service).

Stable Diffusion is a powerful text-to-image generation model based on the Latent Diffusion Model (LDM). It was developed by CompVis, LMU, and RunwayML. The architecture of Stable Diffusion is designed to reduce memory usage and computation time by applying the diffusion process over a lower-dimensional latent space rather than the high-dimensional image space.

**Key Components**

**Variational Autoencoder (VAE)**

The VAE consists of an encoder and a decoder. During training, the encoder converts an image into a low-dimensional latent representation, which is then used for the forward diffusion process (turning an image into noise). The decoder transforms the low-dimensional representation back into an image during the reverse diffusion process (converting noise into an image).

**U-Net**

U-Net is a convolutional neural network that predicts the denoised image representation of noisy latents. It consists of an encoder with 12 blocks, a middle block, and a decoder with 12 blocks. The architecture includes down-sampling and up-sampling convolution layers, ResNet layers, and Vision Transformers (ViTs)[**1**](https://www.bing.com/ck/a?!&&p=976d36b60cff28618ed0e96c268a1a5265a35199690fa3cc26a6206e505bfd99JmltdHM9MTc2MDQ4NjQwMA&ptn=3&ver=2&hsh=4&fclid=329d357b-8a28-62a8-364a-23128b36631d&u=a1aHR0cHM6Ly93d3cudHV0b3JpYWxzcG9pbnQuY29tL3N0YWJsZS1kaWZmdXNpb24vc3RhYmxlLWRpZmZ1c2lvbi1hcmNoaXRlY3R1cmUuaHRt&ntb=1).

**Text Encoder**

The text encoder is a transformer-based model that transforms input tokens into latent text embeddings. Stable Diffusion uses the pre-trained CLIP text encoder to generate embeddings corresponding to the input text. These embeddings guide the denoising process during U-Net's training[**1**](https://www.bing.com/ck/a?!&&p=976d36b60cff28618ed0e96c268a1a5265a35199690fa3cc26a6206e505bfd99JmltdHM9MTc2MDQ4NjQwMA&ptn=3&ver=2&hsh=4&fclid=329d357b-8a28-62a8-364a-23128b36631d&u=a1aHR0cHM6Ly93d3cudHV0b3JpYWxzcG9pbnQuY29tL3N0YWJsZS1kaWZmdXNpb24vc3RhYmxlLWRpZmZ1c2lvbi1hcmNoaXRlY3R1cmUuaHRt&ntb=1).

**Advanced Features in Stable Diffusion 3**

Stable Diffusion 3, released by Stability AI, introduces several enhancements, including flow matching and timestamp sampling techniques. These techniques optimize the diffusion process, ensuring the shortest and most efficient path to generate high-quality images **n**.

**Flow Matching**

Flow matching is a training objective that regresses onto a target vector field to produce a desired probability path. This method transforms a simple distribution into a complex one by applying a series of invertible transformation functions[**2**](https://www.bing.com/ck/a?!&&p=a12a7bb1b43192b67b22c30f05c7b7f7a07fbc469ea4019d2ec812b565f247efJmltdHM9MTc2MDQ4NjQwMA&ptn=3&ver=2&hsh=4&fclid=329d357b-8a28-62a8-364a-23128b36631d&u=a1aHR0cHM6Ly9sZWFybm9wZW5jdi5jb20vc3RhYmxlLWRpZmZ1c2lvbi0zLw&ntb=1).

**Timestamp Sampling**

Timestamp sampling assigns greater importance to intermediate timestamps, which are more significant than those at the beginning or end. This technique uses Logit-Normal Sampling to achieve optimal results

This code snippet demonstrates how to set up the pipeline and generate an image based on a text prompt[**2**](https://www.bing.com/ck/a?!&&p=a12a7bb1b43192b67b22c30f05c7b7f7a07fbc469ea4019d2ec812b565f247efJmltdHM9MTc2MDQ4NjQwMA&ptn=3&ver=2&hsh=4&fclid=329d357b-8a28-62a8-364a-23128b36631d&u=a1aHR0cHM6Ly9sZWFybm9wZW5jdi5jb20vc3RhYmxlLWRpZmZ1c2lvbi0zLw&ntb=1).

**Conclusion**

Stable Diffusion's architecture, based on the Latent Diffusion Model, efficiently generates high-quality images from text prompts. The use of VAE, U-Net, and text encoders, along with advanced techniques like flow matching and timestamp sampling, makes it a powerful tool for text-to-image generation.

**Diffusion Models**

**Core Principle**

Diffusion models simulate a **two-phase process**:

* **Forward Process**: Gradually adds Gaussian noise to data over time.
* **Reverse Process**: Learns to remove noise step-by-step to reconstruct the original data.

This mimics physical diffusion (e.g., ink spreading in water) and is modeled as a **Markov chain**.

**Architecture**

* **U-Net** is commonly used, with skip connections to preserve spatial details.
* **DiT (Diffusion Transformer)** is a newer variant using transformer blocks instead of convolutions.

**Training**

* Trains a neural network to predict the noise added at each step.
* Uses **Mean Squared Error (MSE)** loss between predicted and actual noise.
* Optimization via gradient descent and backpropagation.

**Applications**

* **Image generation** (e.g., Stable Diffusion, DALL·E 2).
* **Text-to-image synthesis**.
* **Audio generation**.
* **Molecular design**.
* **Image inpainting and super-resolution**.

**Strengths**

* **High-quality outputs** with fine details.
* **Stable training** (no adversarial instability).
* **Flexible architecture** (can use CNNs or Transformers).

**Weaknesses**

* **Slow sampling** due to iterative denoising.
* **High computational cost**.
* **Complex implementation**.

**🔹 2. Generative Adversarial Networks (GANs) Core Principle**

GANs consist of two networks:

* **Generator (G)**: Creates fake data from noise.
* **Discriminator (D)**: Classifies data as real or fake.

They train via a **minimax game** where G tries to fool D, and D tries to catch G.

**Architecture**

* **Generator**: Often uses transposed convolutions.
* **Discriminator**: Uses CNNs to classify real vs. fake.
* Variants include **DCGAN**, **CGAN**, **WGAN**, **SRGAN**, etc.

**Training**

* Uses **Binary Cross-Entropy** or **Wasserstein loss**.
* Training is adversarial and can be unstable.
* Requires careful tuning of hyperparameters.

**Applications**

* **Image synthesis** and enhancement.
* **Style transfer**.
* **Text-to-image generation**.
* **Data augmentation**.
* **Video generation**.

**Strengths**

* **Fast generation**.
* **High realism** in outputs.
* **Versatile** across domains.

**Weaknesses**

* **Training instability** (e.g., mode collapse).
* **Sensitive to hyperparameters**.
* **Limited control over output diversity**.

**🔹 3. Transformers**

**Core Principle**

Transformers use **self-attention** to model relationships in sequences. They process entire sequences **in parallel**, unlike RNNs.

**Architecture**

* **Encoder-Decoder** structure:
  + **Encoder**: Processes input sequence.
  + **Decoder**: Generates output sequence.
* Key components:
  + **Multi-head attention**
  + **Positional encoding**
  + **Feed-forward layers**
  + **Residual connections and layer normalization**

**Training**

* Trained on large datasets using **next-token prediction**.
* Uses **cross-entropy loss**.
* Fine-tuned with techniques like **RLHF** (Reinforcement Learning from Human Feedback).

**Applications**

* **Natural Language Processing** (e.g., ChatGPT, BERT).
* **Machine translation**.
* **Text summarization and question answering**.
* **Vision tasks** (e.g., Vision Transformers).
* **Code generation**, **protein folding**, **DNA analysis**.

**Strengths**

* **Handles long-range dependencies**.
* **Highly parallelizable**.
* **Scalable** to large models.
* **Versatile** across modalities (text, image, audio).

**Weaknesses**

* **Data and compute intensive**.
* **May struggle with fine-grained image details**.
* **Requires positional encoding to retain order**.