CS21 - Genereative and Agentic AI

SOS'25 Final Report

July 18, 2025

Name : Manthan Roll no : 24b2511 Mentored by :

1 Introduction to Generative and Agentic AI

1.1 Overview

Generative AI refers to artificial intelligence systems designed to autonomously create new content—such as text, images, music, or code—by learning patterns from existing data. Algorithms in this area can produce realistic outputs, ranging from synthetic photos to coherent essays.

Agentic AI describes systems that do not only generate content, but also act as agents, taking actions, making decisions, and interacting with tools or environments. These agents possess capabilities for planning, memory management, goal-directed reasoning, and tool usage.

Recent years have seen explosive growth in both fields: Generative models such as GPT, DALL·E, and Stable Diffusion have made it possible to rapidly generate creative outputs in seconds. Agentic AI has enabled automation of research, code generation, web browsing, and task planning, giving rise to personal digital assistants and autonomous research agents.

Distinction: Generative AI focuses on content creation; agentic AI emphasizes decision-making, planning, and environmental interaction. Overlap: Many modern systems (e.g., conversational agents) combine both—using generative models within larger agentic frameworks.

2 Tools and Software Used

A variety of tools and environments support work in generative and agentic AI:

2.1 Environments & Tools

Python: Ubiquitous in AI/ML, with broad support for scientific computing, data manipulation, and ML frameworks.

Jupyter Notebooks: Interactive and visual, ideal for exploration and documentation. Google Colab: Cloud-based, GPUs/TPUs for scalable training. RASA: Industry-standard toolkit for building production chatbots with customizable dialogue management.

Evaluation Libraries : ***FID*** (Fréchet Inception Distance): Compares feature distributions of real and synthesized images, with lower scores indicating higher similarity. ***BLEU*** (Bilingual Evaluation Understudy): Measures translation/text-generation faithfulness via n-gram overlaps with human references. Suggested Image: Side-by-side comparison chart or plot showing FID scores for different generative models on benchmark tasks.

3 Generative AI: Concepts and Deep Techniques

3.1 GANs (Generative Adversarial Networks)

Core Idea:
Two neural networks ("Generator" and "Discriminator") compete—a minimax game—where learns

Tool	Purpose
TensorFlow	Deep Learning framework with Keras API for rapid prototyping
HuggingFace Transformers	Pretrained models for NLP (e.g., GPT, BERT) and generative tasks
LangChain & LangGraph	Frameworks for building and deploying AI agents; chaining tools and prompts
RAG (Retrieval-Augmented Generation)	Combines language generation with factual retrieval from external sources

Table 1: Popular Tools and Their Purposes in Generative and Agentic AI

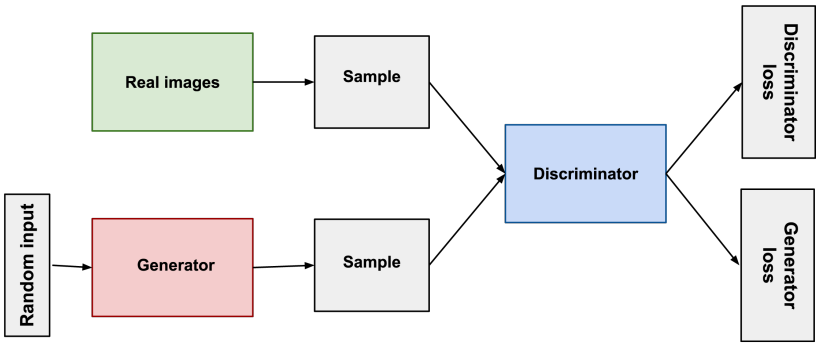

Figure 1: Figure 1: Basic GAN Architecture.

to produce realistic data by attempting to fool , and improves at detecting fakes.
Detailed Architecture:

- Generator: Maps random noise vector to data space. Learns the conditional distribution .
- Discriminator: Classifies inputs as authentic (from training set) or fake (from generator).

Objective Function: In-Depth Training Issues:

- Mode Collapse: Generator finds a few outputs that "fool" ; diversity suffers.
- Vanishing Gradients: If is too strong, stops learning.
- Wasserstein GANs (WGAN): Improve stability by using Earth-Mover distance.
- Suggested Image: Schematics showing Generator and Discriminator with arrows for data and gradient flows (Figure 1: Basic GAN Architecture).

3.2 Diffusion Models

Core Process: Forward Process: Adds small noise steps to make data indistinguishable from pure noise.Reverse Process: Learns to denoise stepwise, reconstructing coherent samples. Equations:
Foward Noise :

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\,\epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

Reverse: Model predicts the original by gradually denoising .

Strengths:
Generates ultra-high resolution, diverse images.Robust against mode collapse. Cutting-edge for text-to-image generation: Stable Diffusion, Denoising Diffusion Probabilistic Models (DDPMs).

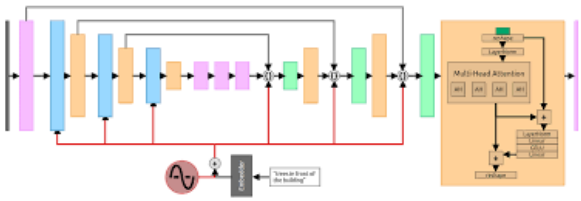

Figure 2: Diffusion Process Illustration

3.3 Transformer-Based Generative Models (Deep Dive)

Self-Attention: Enables model to weigh importance of all previous tokens for each prediction. .

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

Decoder-Only Models: E.g., GPT, optimized for autoregressive text output.

Decoder-Encoder Architectures: E.g., BERT, T5, excel at both generation and understanding.

Layer Normalization, Residuals, Position Encoding:

Maintains stable gradients, preserves spatial/temporal context, enables effective deep stacking.

Prompt Engineering: Techniques for steering LLM output, e.g., few-shot learning, chain-of-thought prompts, system message tuning.

3.4 Computer Vision & Multimodal Generative AI

Super-resolution: Enhances image detail with learned upsampling. *Inpainting*: Predicts and fills missing image regions. *Text-to-Image and Image-to-Image Translation*: Leverages paired/unpaired learning. Suggested Image: Grid of generated faces (from StyleGAN), and sample super-resolution or inpainted photo outputs.

4 Agentic AI: Advanced Concepts

4.1 AI Agents Fundamentals

Autonomy: Acts without continuous human intervention. *Reactivity & Proactivity*: Responds to changes and initiates actions toward goals. *Embodied/Virtual Agents*: Robots, software bots, digital humans.

Reasoning, Planning, Memory Deep Dive • Classical Reasoning: Symbolic/logic-based, e.g., search algorithms (A*, alpha-beta pruning). • Neuro-Symbolic Approaches: Hybrid of rule-based and neural representation. • Planners: • Model-based: Search trees, Monte Carlo Tree Search. • Model-free: Reinforcement learning, policy gradients. • Memory: • Short-term (working): Recent context retention. • Long-term: Episodic (event logs), semantic (facts/knowledge). • Vector Stores: Embedding storage for context retrieval (FAISS, Pinecone). Suggested Image: Diagram showing agent loop: perception → reasoning → action → memory update.

4.2 Agentic Frameworks & Architectures

• LangChain: Modular building blocks for chaining LLM "tools" (APIs, databases), sophisticated context memory. • AutoGPT: Autonomous goal decomposition, planning, action loops, "thought" logs. • Multi-agent Collaborations: Teams of agents solving complex, distributed tasks (simulation games, research, collaborative filtering). There are several more Agentic frameworks like LangGraph,Crew AI and many more

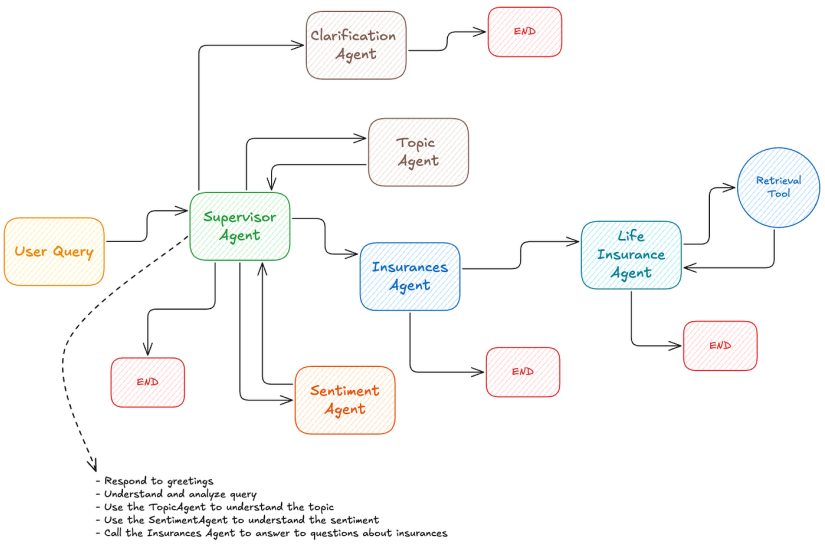

Figure 3: Agentic AI Workflow

5 Essential Technical Terms and Concepts (Expanded)

Term	**Deep Explanation**
Prompt Engineering	Crafting queries to elicit specific, optimal responses, leveraging structure and context.
RAG	Fuses fast retrieval (vector search) + generation (LLM completion) for up-to-date answers.
Embeddings	Multi-dimensional vectors learned from data, encode semantic similarity, clustering power.
VectorStores	Databases optimized for dense vector indexing & similarity search.
Memory Types	Agent memory segregation: recency, relevance, persistence for robust context management.
Tokenization	Splitting input into subword units, optimizing for rare and frequent words efficiently
Attention Mechanism & LangGraph	Network structures that dynamically focus on crucial input parts for each output.
ReAct Prompting	Repeated loop of reasoning/acting, often alternating tool use (e.g., search, calculation).

Table 2: Popular Tools and Their Purposes in Generative and Agentic AI

6 Applications and Hands-on Tasks

1. *GANs on CIFAR-10*: Image synthesis and studies showing the effect of generator learning rate.
2. *Conditional GANs*: Class-specific image generation, such as only "airplanes."
3. *Diffusion Models for Custom Art*: Train on small datasets and analyze the effect of noise schedules and denoising steps.
4. *Chatbot with RASA*: Combine intent classification, slot filling, and dialogue tracking for FAQ and transactional tasks.
5. *LangChain Agents and Wikipedia API*: A multi-hop question-answering agent with interpretability through the agent's internal thoughts.

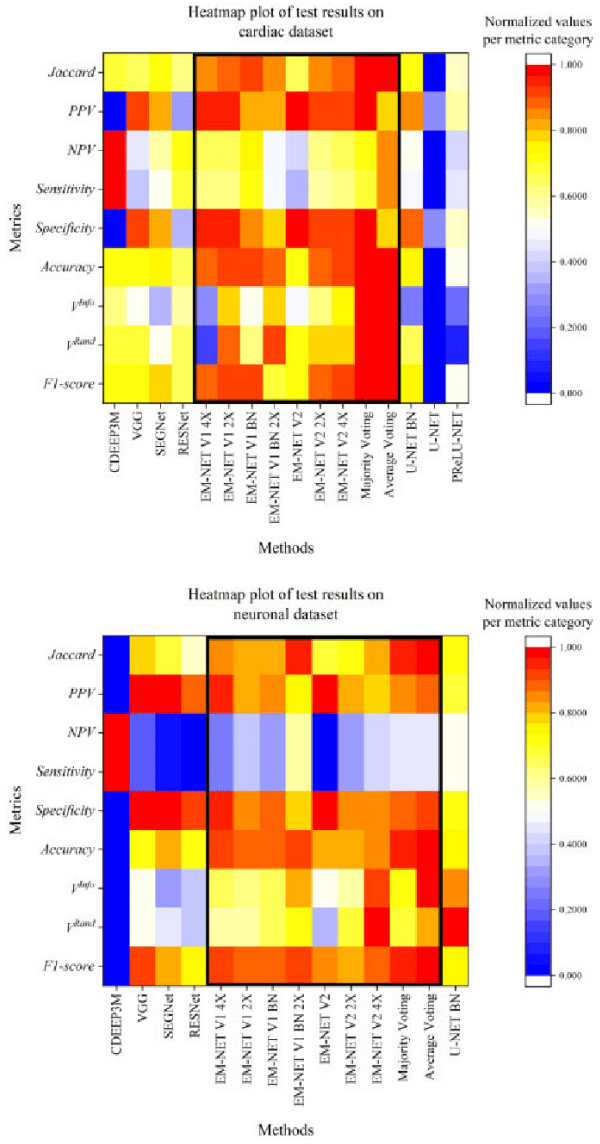

Figure 4: A heatmap or bar plot showing evaluation metric trends across different models and datasets.

6. Multi-Agent Task Allocation: Agents simulate negotiation and coordination in collaborative problem-solving tasks.
7. **Evaluation**:
8. *FID*: Compare Inception-v3 embeddings of real and generated samples.
9. *BLEU*: Unpack n-gram overlap and apply a penalty for brevity and disfluency.
10. *Perceptual similarity*: Use learned perceptual image patch similarity (LPIPS) for visual models.

7 Project Report: Retrieval-Augmented Generation (RAG) using Groq LLM

7.1 Project Overview

This project demonstrates a Retrieval-Augmented Generation (RAG) pipeline that allows users to query a knowledge base (built from PDF documents) and receive answers generated by a Large Lan-

guage Model (LLM). The RAG pipeline combines dense document retrieval with a powerful generative model to provide context-aware answers.

7.2 Objectives & Technologies

Extract text from a PDF document.Chunk and vectorize the extracted content.Build a FAISS-based vector store.Perform semantic search to retrieve relevant chunks.Generate answers using a Groq-hosted LLM based on the retrieved context.

LangChain: For building the text splitting and RAG pipeline.

FAISS: For fast similarity search on embeddings.

HuggingFace Embeddings: Specifically the all-MiniLM-L6-v2 model.

Groq API: LLM API for generating answers using models like LLaMA-3.

Streamlit: For front-end integration (not fully shown in provided cells).

PyPDF2: To extract raw text from PDF files.

7.3 Methodology & Workflow

A custom function load_and_chunk_pdf(path) is created which:
Reads PDF pages using PyPDF2.PdfReader.Extracts and aggregates raw text.Splits the text into manageable chunks (500 chars with 50 overlap) using RecursiveCharacterTextSplitter.

Vector Store Construction : The function create_faiss_index(chunks):Wraps text chunks into Document objects.Converts them into vector embeddings using HuggingFace's all-MiniLM-L6-v2.Stores them in a FAISS index for fast retrieval.
Retrieval Function retrieve_docs(query, db, k=15) performs similarity search to fetch top-k relevant document chunks based on user queries.
LLM-Based Answer Generation generate_answer_groq(query, context, model="llama3-8b-8192") constructs a prompt with retrieved context and sends it to Groq's LLM API.
Returns the generated answer.

Note: The Groq API key is expected to be added by the user manually ("GROQ_API_KEY" placeholder is present).

Visit Project GitHub Repo

8 References

References

1. Goodfellow, I., et al. "Generative Adversarial Nets." NeurIPS 2014
2. Ho, J., et al. "Denoising Diffusion Probabilistic Models." NeurIPS 2020
3. Vaswani, A., et al. "Attention is All You Need." NeurIPS 2017
4. HuggingFace Transformers Documentation
5. TensorFlow and PyTorch Official Docs
6. LangChain Framework Docs

7. OpenAI GPT-3 Paper
8. Coursera: Generative Deep Learning Specialization
9. Towards Data Science: Blogs on GANs, diffusion, agentic AI
10. Two Minute Papers (YouTube)
11. RASA Documentation
12. FAISS/Pinecone Vector Store Docs
13. LPIPS Perceptual Similarity Measure

End of Report