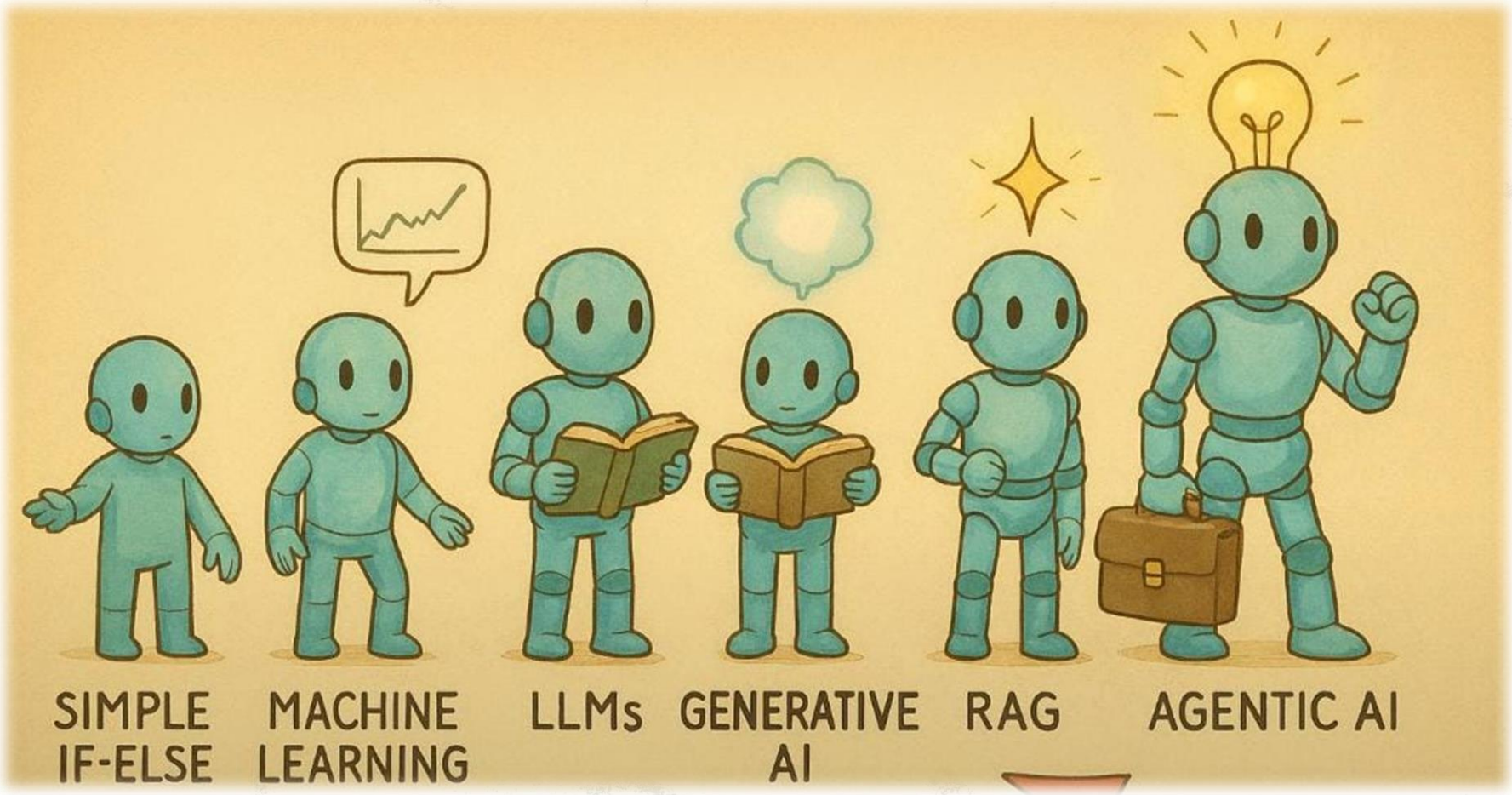
Scope and opportunities in the areas of

Artificial Intelligence - Machine Learning to Agentic AI

Dr. Sarwan Singh

Scientist – D, NIELIT Chandigarh

Journey from Traditional Programming to Agentic AI

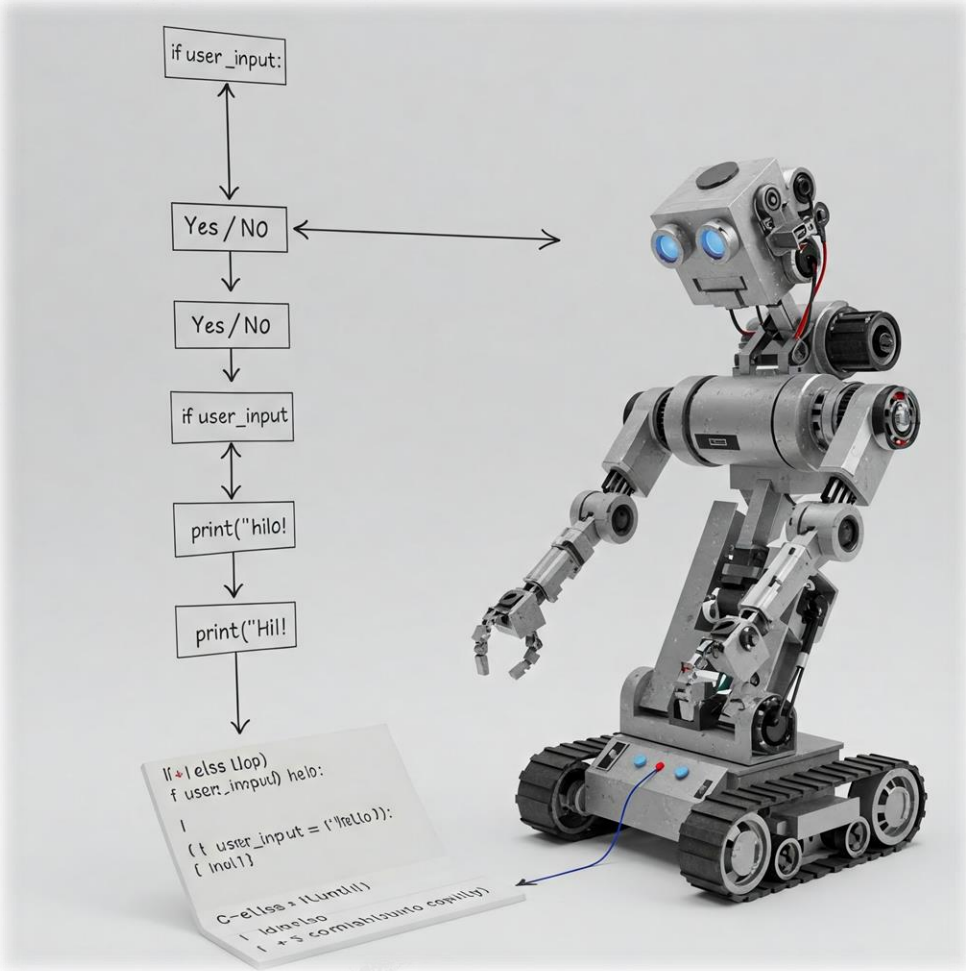


References

- IIT Gandhinagar eMasters course
- Website :

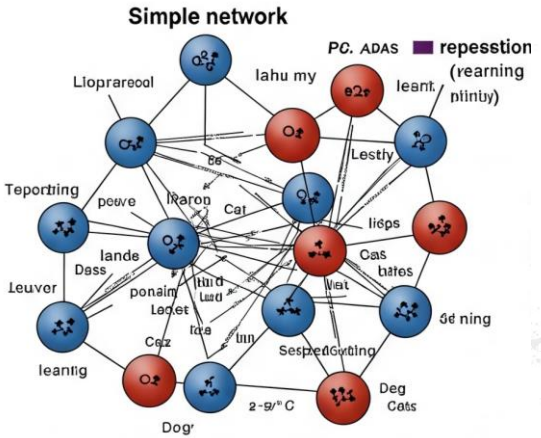
Traditional Programming (Imperative & Rule-Based) (1950s–Present)

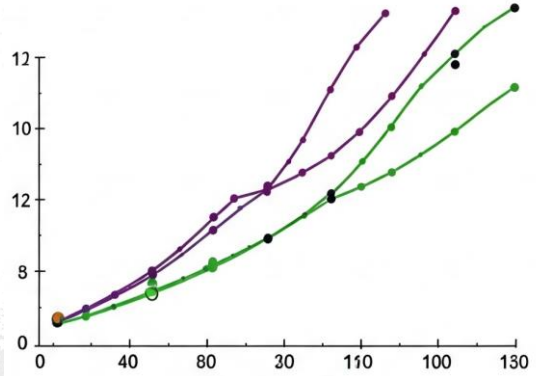
- **Description**: Explicit logic using if-else, loops, and functions. Developers write rules to process inputs and produce outputs.
- **Era**: Dominant since the dawn of computing (e.g., Fortran, C, Python).
- **Key Idea**: "Tell the computer exactly what to do.“



Machine Learning (ML) (1980s–Present)

- **Description**: Systems learn patterns from data instead of relying on hardcoded rules. Includes supervised/unsupervised learning (e.g., regression, decision trees).
- **Key Advance**: Let algorithms *learn* rules from data rather than requiring explicit programming.
- **Milestones**:
  - 1986: Backpropagation for neural networks.
  - 1990s: SVMs, Random Forests.
  - 2010s: Rise of deep learning (CNNs, RNNs).



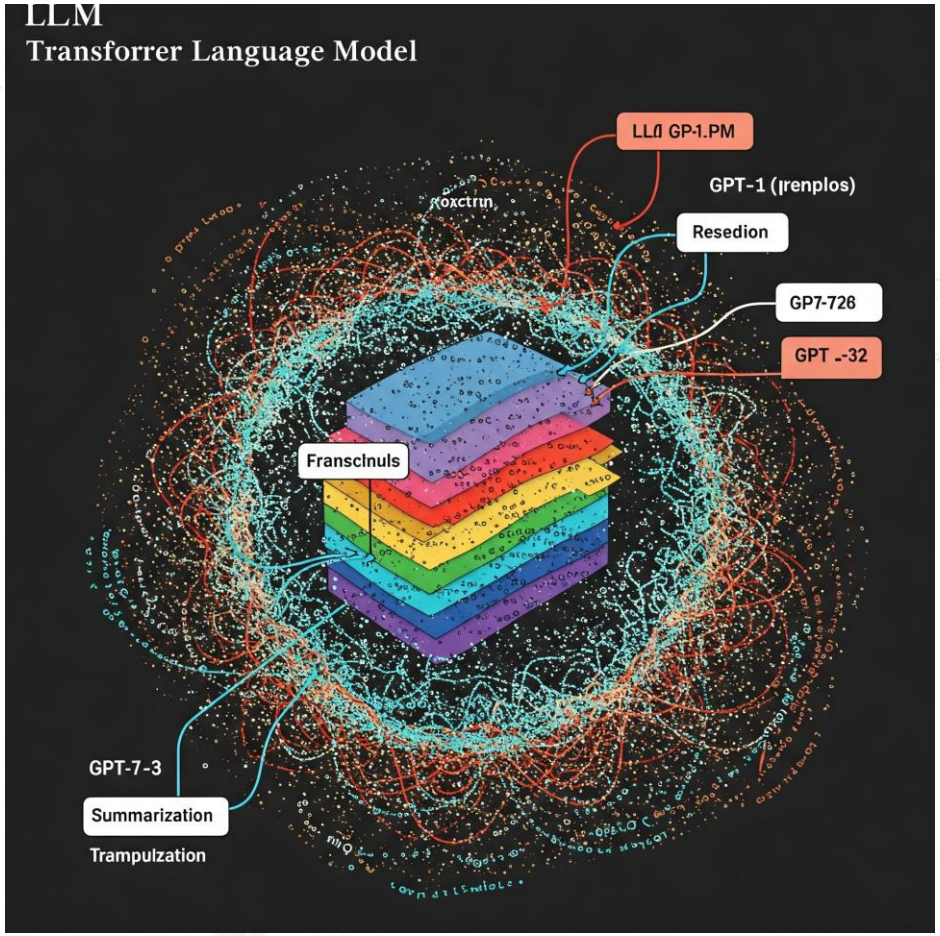


Generative AI (2014–Present)

- **Description**: Models that generate new content (text, images, etc.) by learning data distributions. Early examples: GANs (2014), VAEs.
- **Key Idea**: Create *new* data similar to training data.
- **Applications**: Image synthesis (e.g., StyleGAN), early text generation (e.g., GPT-1).

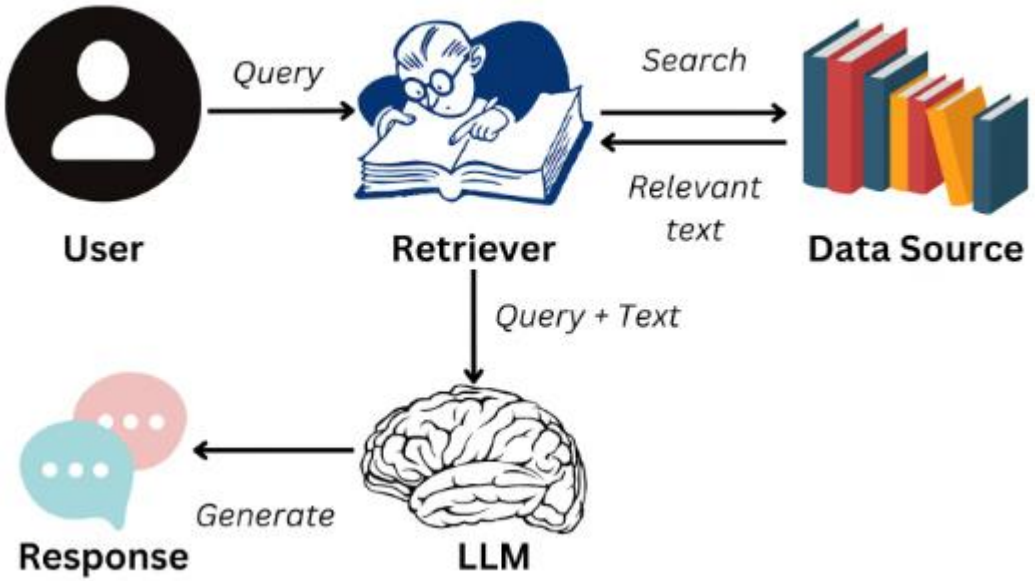
Large Language Models (LLMs) (2018–Present)

- **Description**: Transformer-based models (e.g., GPT, BERT) trained on massive text data for tasks like translation, summarization, and Q&A.
- **Milestones**:
  - 2017: Transformer architecture (Vaswani et al.).
  - 2018: GPT-1, BERT.
  - 2020: GPT-3 (175B parameters).
- **Shift**: Few-shot/zero-shot learning without task-specific training.



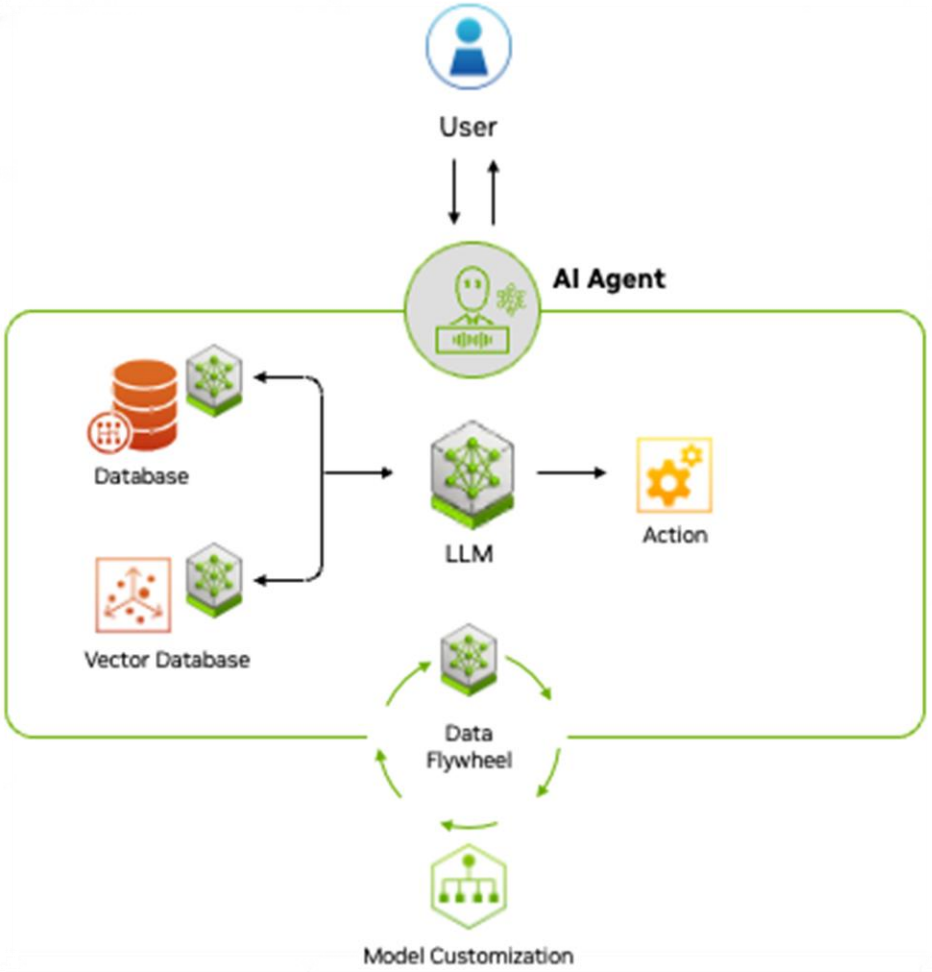
Retrieval-Augmented Generation (RAG) (2020–Present)

- **Description**: Combines LLMs with external knowledge retrieval (e.g., databases, search) to improve accuracy and reduce hallucinations.
- **Key Idea**: Ground LLM outputs in real-time, verifiable data.
- **Example**: ChatGPT + web search for up-to-date answers.



Agentic AI (2023–Present)

- **Description**: AI "agents" that autonomously plan, act, and iteratively improve (e.g., AutoGPT, BabyAGI). Uses LLMs + tools (APIs, code execution).
- **Key Idea**: *Autonomous* goal-directed behavior with memory/feedback loops.
- **Features**: Self-prompting, multi-step reasoning, tool use.



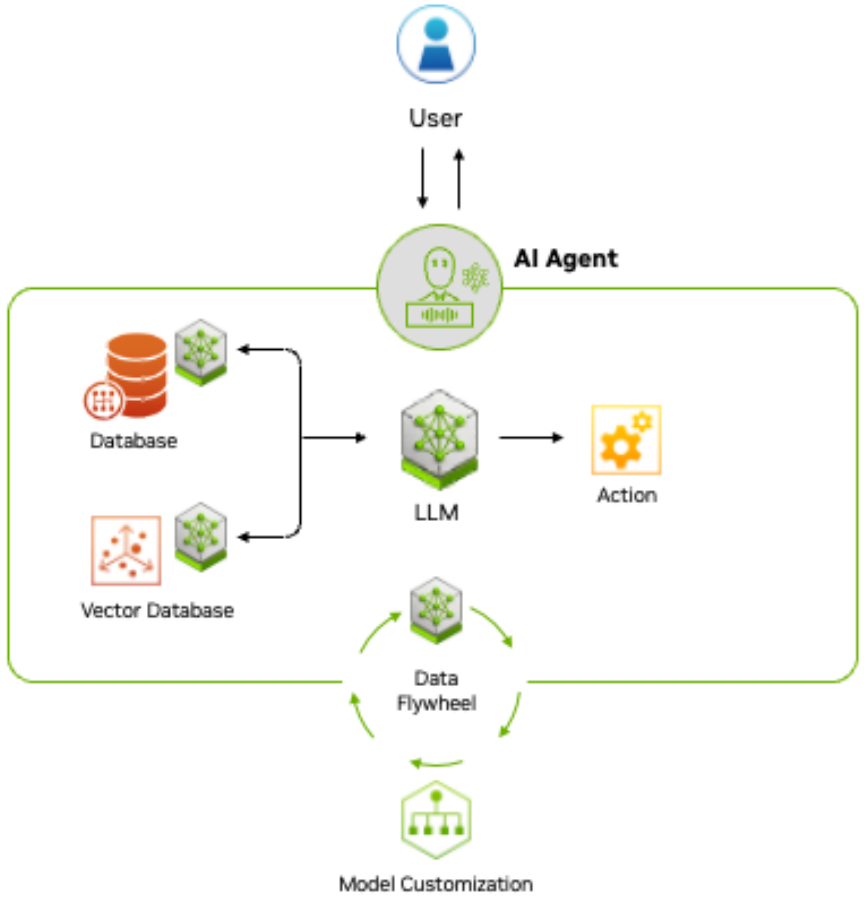
Timeline Summary:

- **1950s–1980s**: Rule-based programming.
- **1980s–2010s**: Machine learning (statistical models → deep learning).
- **2010s**: Generative AI (GANs, early transformers).
- **2018–2022**: LLMs (GPT, BERT, scaling laws).
- **2020–2023**: RAG (dynamic knowledge integration).
- **2023–Future**: Agentic AI (autonomous systems).
- Each phase builds on prior advances, with LLMs acting as the foundation for modern generative AI, RAG, and agents.
- The trend moves from *explicit programming → learning from data → autonomous reasoning*.

Agentic AI

Agenda

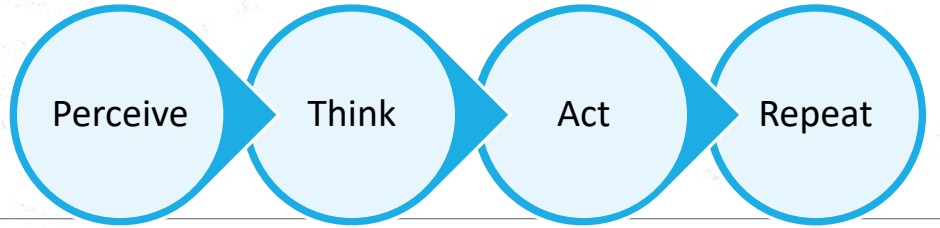
- What are agents?
- What are agentic frameworks?
- Architecture & Flow of agentic frameworks
- Factors to be considered for Agentic projects / platforms / products
- Productionizing the Agentic framework implementation
- Use cases
- Tech Stack
- Guidelines & best practices / patterns
- Configurations at each level
- High level coding guidelines
- A2A and MCP
- Demo



How does Agentic AI works

- **Perceive**: AI agents gather and process data from various sources, such as sensors, databases and digital interfaces. This involves extracting meaningful features, recognizing objects or identifying relevant entities in the environment.
- **Reason**: A large language model (LLM) acts as the orchestrator, or reasoning engine, that understands tasks, generates solutions and coordinates specialized models for specific functions like content creation, visual processing or recommendation systems. This step uses techniques like retrieval-augmented generation (RAG) to access proprietary data sources and deliver accurate, relevant outputs.
- **Act**: By integrating with external tools and software via application programming interfaces, agentic AI can quickly execute tasks based on the plans it has formulated. Guardrails can be built into AI agents to help ensure they execute tasks correctly. For example, a customer service AI agent may be able to process claims up to a certain amount, while claims above the amount would have to be approved by a human.
- **Learn**: Agentic AI continuously improves through a feedback loop, or "data flywheel," where the data generated from its interactions is fed into the system to enhance models. This ability to adapt and become more effective over time offers businesses a powerful tool for driving better decision-making and operational efficiency.

What are Agents



- Essentially AI agents & agentic frameworks build a wrapper on top of LLMs to build context, chats, custom code, configurations, various integrations, web surfing/scraping, workflows, document scanning, etc. all programming control like other standard language frameworks - example SPRING, LARAVEL, etc.
- Simple reflex agents – react directly to perceptions.
- Model-based agents – maintain internal state/history.
- Goal-based agents – act toward specific goals.
- Utility-based agents – aim for the best possible outcome.
- Learning agents – improve their performance over time.

What are Agentic Frameworks

An agentic framework provides the infrastructure and logic that allows an AI agent to:

- Understand and decompose goals
- Make decisions across multiple steps
- Use tools (like web search, databases, or APIs)
- Track memory or context
- Learn from past actions

Core Components in Agentic Frameworks

- Agent Logic – How the agent thinks (planning, reasoning, learning)
- Memory – Stores past actions, decisions, or conversations
- Tooling – Allows the agent to use external tools (e.g., Google search, file access)
- Planning/Execution – Can break goals into smaller tasks and execute them
- Feedback Loop – Reflects on outcomes and adjusts next steps

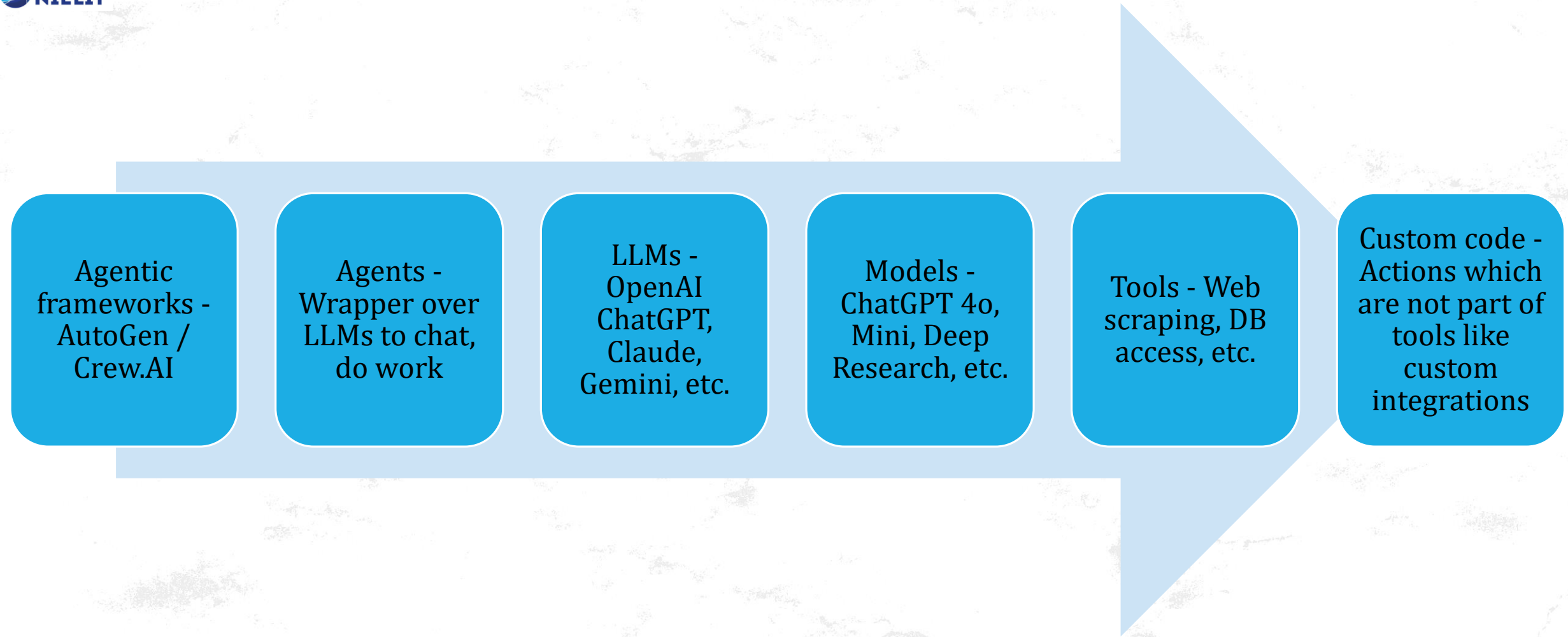
Agentic Framework Examples

- LangChain
- Auto-GPT
- CrewAI
- BabyAGI
- Autogen

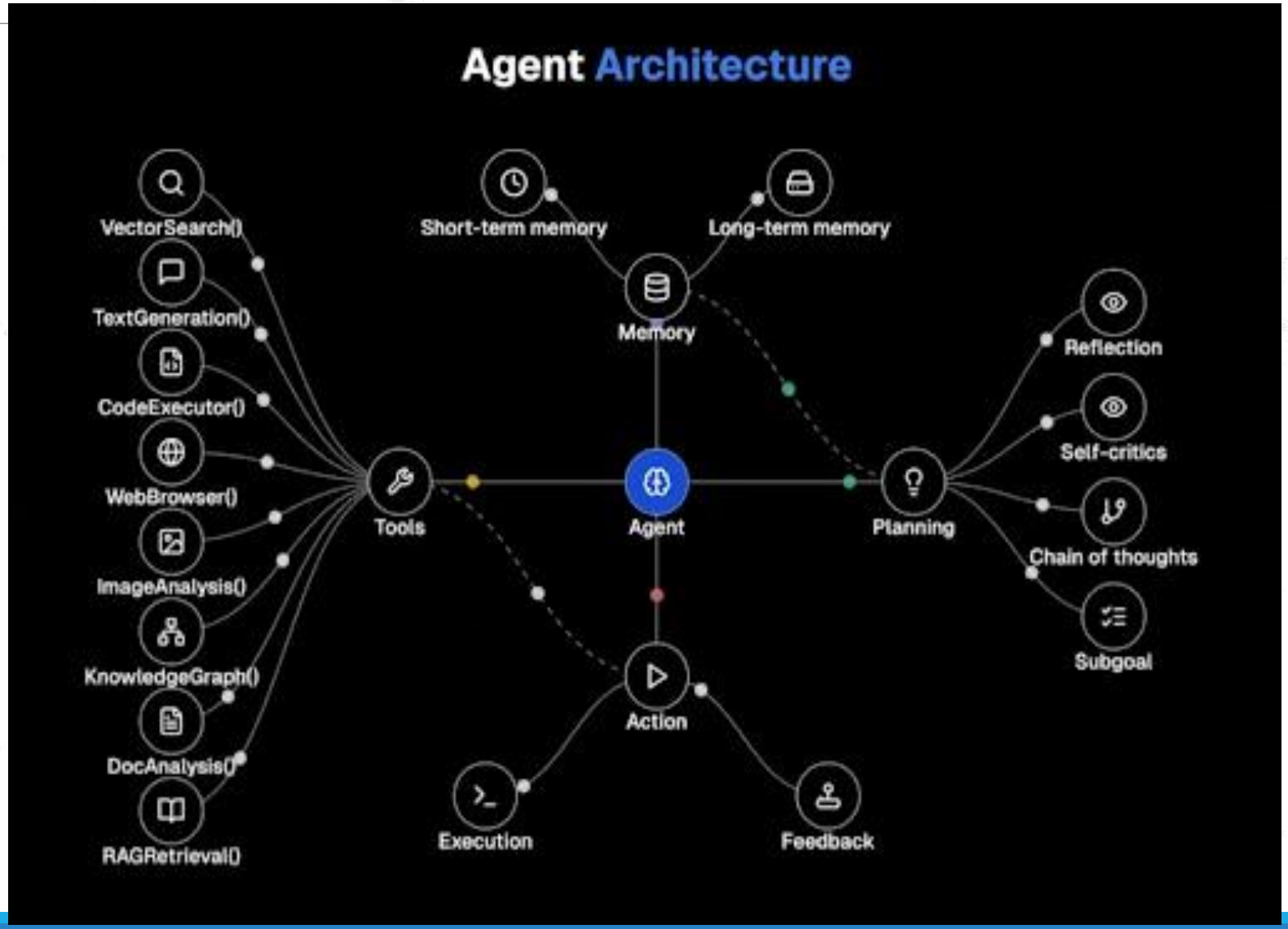
Architecture & Flow of agentic frameworks

- Agentic frameworks - AutoGen / Crew.AI
  - Agents - Wrapper over LLMs to chat, do work
    - LLMs - OpenAI ChatGPT, Claude, Gemini, etc.
      - Models - ChatGPT 4o, Mini, Deep Research, etc.
        - Tools - Web scraping, DB access, etc.
          - Custom code - Actions which are not part of tools like custom integrations

Architecture & Flow of agentic frameworks



Agent Architecture



Factors to be considered for Agentic projects / platforms / products

- Temporary vs. permanent chat
- Memory
- Reflection / context
- Chain of thoughts
- Creativity
- Hallucinations
- Configurations
- And so on

Use case

Productionizing the Agentic framework implementation

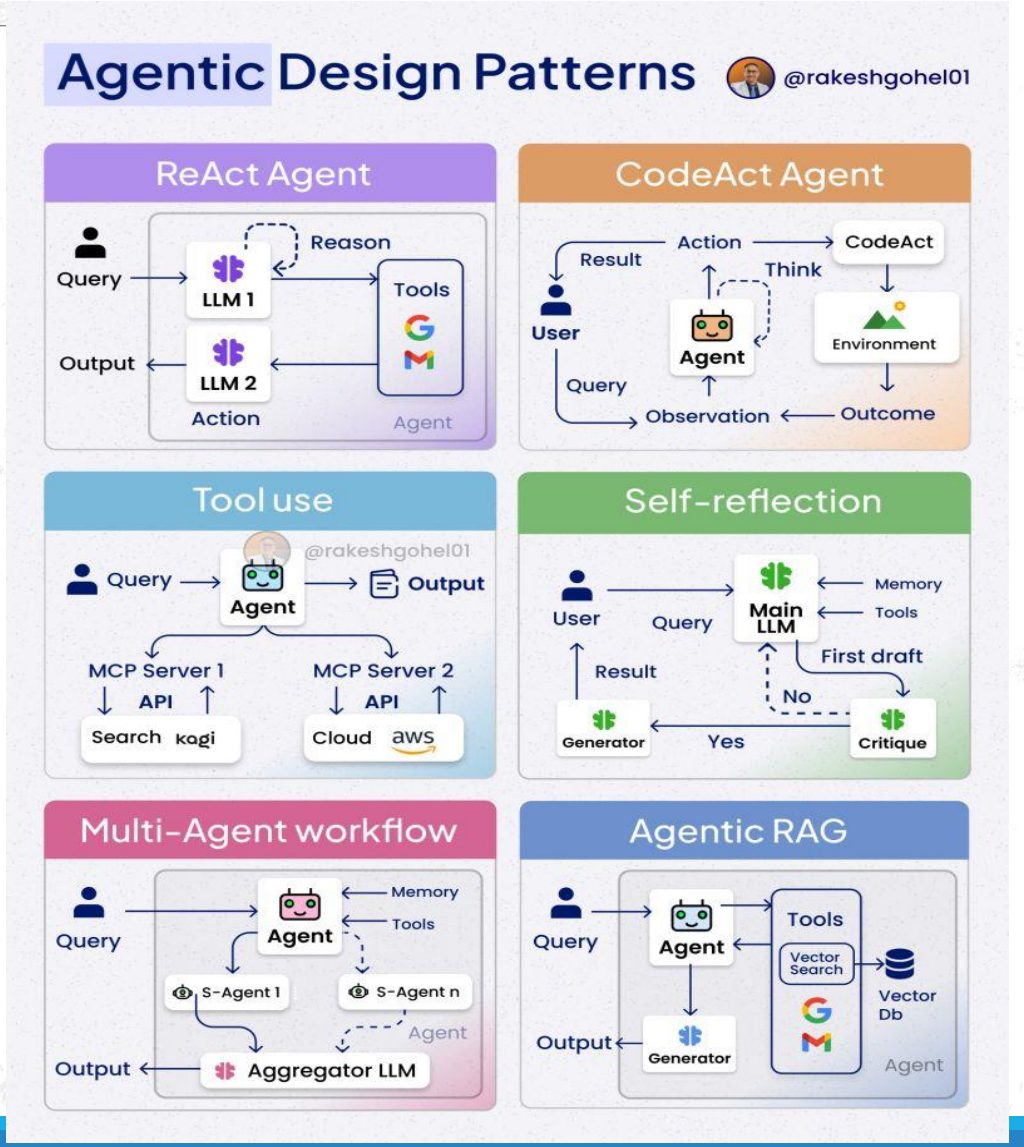
- Licensing
- Deployment
- Dockerization
- Kubernetes
- Scaling
- Code obfuscation
- Packaging
- Framework selection

Tech Stack

- N8n - Workflow and Automation
- Agentic framework - AutoGen/Crew.AI/LangGraph
- Python
- LLMs - ChatGPT/Gemini/Grok/Claude/etc.
- Models - 4o, mini, etc.
- Agentic framework tools
- LangSmith - Debugging & Observability for Agents

Guidelines & best practices / patterns

- Hierarchical vs. non hierarchical agents
- Async integrations
- Control via congratulations that's important
- OWASP Top 10 for LLMs
- Use pipeline approach for your work
- https://www.linkedin.com/posts/rakeshgohel01_if-you-are-building-ai-agents-utilize-these-activity-7311001517048623104-qoEL



Configuration at each level

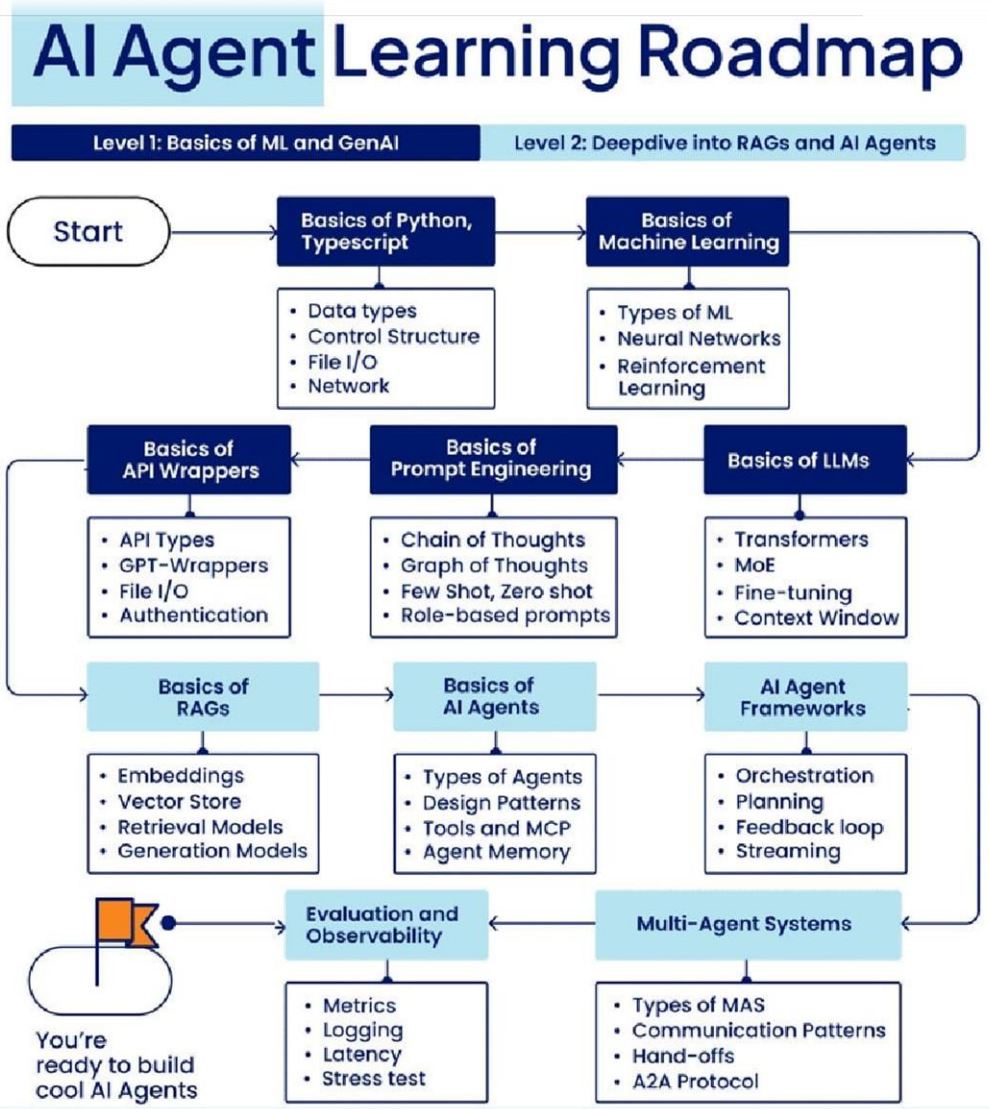
- Temperature
- Heat
- Penalty
- Max Tokens

Demo

- Multi - agent system for web scraping and data analysis
- Get title of www.economictimes.com: https://colab.research.google.com/drive/1IulbXupl_G4crIeh74qocrtwtsrt9zQ0?usp=sharing
- CONTROLLER -> WEB SCRAPER -> DATA ANALYZER FROM WEB -> SOME INFERENCE
- Medical Imaging agent
- Travel planning Agent - Gemini and Serp
- Email automations for data science jobs - N8N
- Deep Research agents - N8N
- Health fitness planner agent - gemini and serp
- Youtube transcription agent - N8N

AI Agent Learning Roadmap

- Basics of Python, Typescript
- Basics of Machine Learning
- Basics of LLMs
- Basics of Prompt Engineering
- Basics of API Wrappers
- Basics of RAGs
- Basics of AI Agents
- AI Agent Framework
- Multi Agent Systems



Level 1: Basics of ML and GenAI

Programming Foundations

- **Python & TypeScript**: Core programming languages for AI development
  - Data types: Primitives, collections, custom types
  - Control structures: Conditionals, loops, error handling
  - File I/O: Reading/writing files, working with different formats
  - Network: HTTP requests, sockets, APIs

Machine Learning Fundamentals

- **Types of ML**:
  - Supervised learning (classification, regression)
  - Unsupervised learning (clustering, dimensionality reduction)
  - Reinforcement learning (agents, rewards, environments)
- **Neural Networks**:
  - Basic architecture (input/hidden/output layers)
  - Activation functions, backpropagation
  - Common architectures (CNNs, RNNs)

API Wrappers

- **API Types**: REST, GraphQL, gRPC
- **GPT-Wrappers**: OpenAI API, Anthropic, other LLM providers
- **Authentication**: API keys, OAuth, token management

Level 2: Deep Dive into RAGs and AI Agents

Prompt Engineering

- **Chain of Thought**: Breaking down complex problems step-by-step
- **Graph of Thought**: Non-linear reasoning approaches
- **Few-shot/Zero-shot**: Providing examples vs. no examples
- **Role-based prompts**: Assigning personas to guide responses

Large Language Models (LLMs)

- **Transformers**: Attention mechanisms, encoder-decoder architecture
- **Mixture of Experts (MoE)**: Specialized model components
- **Fine-tuning**: Adapting base models to specific tasks
- **Context Window**: Managing input/output length limitations

Level 2: Deep Dive into RAGs and AI Agents

Retrieval-Augmented Generation (RAG)

- **Embeddings**: Text representation (Word2Vec, BERT, etc.)
- **Vector Stores**: Pinecone, Weaviate, FAISS
- **Retrieval Models**: Semantic search, hybrid search
- **Generation Models**: Integrating retrieval with generation

Level 2: Deep Dive into RAGs and AI Agents

AI Agent Fundamentals

- **Types of Agents**:
  - Reactive, deliberative, utility-based
  - Conversational, autonomous, hybrid
- **Design Patterns**:
  - ReAct, Plan-and-Execute, Reflexion
- **Tools and MCP** (Memory, Control, Perception):
  - Tool use, memory management, perception systems
- **Agent Memory**:
  - Short-term vs. long-term
  - Recall mechanisms

Level 2: Deep Dive into RAGs and AI Agents

AI Agent Frameworks

- **Orchestration**: LangChain, LlamaIndex, Semantic Kernel
- **Planning**: Task decomposition, goal setting
- **Feedback loops**: Self-correction, human-in-the-loop
- **Streaming**: Real-time processing and responses

Level 2: Deep Dive into RAGs and AI Agents

Evaluation and Observability

- **Metrics**:
  - Accuracy, precision, recall
  - Latency, cost, token usage
- **Logging**: Conversation history, decision tracking
- **Stress testing**: Load testing, edge cases

Multi-Agent Systems

- **Types of MAS**:
  - Cooperative, competitive, mixed
  - Hierarchical, decentralized
- **Communication Patterns**:
  - Direct messaging, blackboard, publish-subscribe
- **Hand-offs**: Task transfer protocols
- **Agent-to-Agent (A2A) Protocols**:
  - Standardized communication formats
  - Conflict resolution

Learning Progression

This roadmap progresses from fundamental programming and ML concepts through specialized AI agent development. The path:

- Build programming and ML foundations
- Master prompt engineering and LLM concepts
- Develop RAG systems for knowledge-enhanced generation
- Design and implement autonomous agents
- Scale to multi-agent systems with complex interactions