Learning Systems and Autonomous Agents

Workshop Kick-Off

2025-2026

5.11.2025

MTA

Dr. George Kour

Agenda

- Motivation
- Workshop Logistics and requirements
- Deeper into AI agents
- Projects examples

The AI revolution in Software development

- The AI represents the most significant transformation in computing since programmable computers.
- Enabling systems whose behavior emerges from data rather than explicit programming language instructions.
- This transformation requires new engineering foundations because traditional software engineering principles cannot address systems that learn and adapt based on experience.
- The new software has a wider capability, require different design and infrastructure.

AI Agents

ContextForge
MCP Gateway

crewai

Agents can be defined as software components that can reason, plan and interact with its environment to solve specific problem or meet a specific need.

They are also capable of collaborating and communicating with other agents in a more complex task.

Building agents among other things, requires unique knowledge different than standard software development:

- Some familiarity with machine learning and LLMs
- New programming languages, platforms and frameworks. E.g., MCP, A2A, Langraph, crewAI.
- New programming concepts and design patterns. E.g., Function calling, reflection, etc.
- New operations: like optimization, observability, etc.
- New non-functional unique requirements domains: like autonomity and reliability, etc.
- A hole set of new problems: hillucination, safety, etc.

Hugging Face

LiteLLM

Workshop Logistics

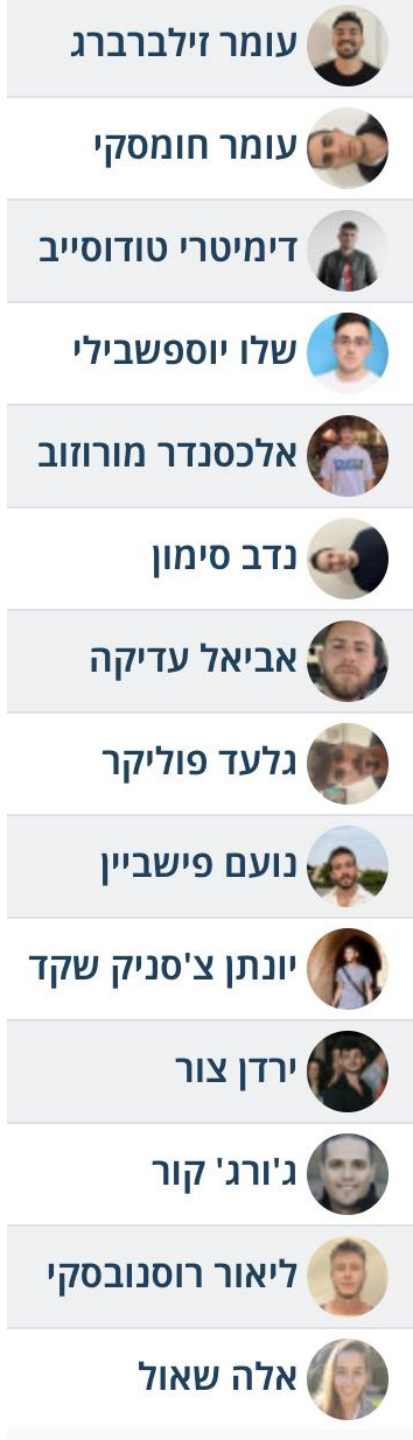
& requirements

Introduction

- Workshop Goal: Define and implement a large-scale project.
- Why? Get a practical engineering/Development Handsons on experience
- 2 Semesters Course. 6 Credit Points.
- Language = English!
  - Documents
  - Presentations
  - Communication
- Project Types:
  - Applicative
  - Infrastructure/Research

Teams

- Omer Zilberger, Gilad Poliker, Lior Rosnovski
- Nadav Simon, Shalev Yosefashvili, Dima Todasiev
- Noam Fishbain, Omer Homski
- Alexander Morozov, Yarden Tsur
- Ella Shaul, Yonathan Shesnik, Aviel Adika



Timeline

- Semester A:
  - Exploring a specific need/problem/Question.
  - Lectures – will define later.
  - Creating project website.
  - Report the project type.
  - Project Description (2-3 pages): Describing the Need/Problem/Importance.
  - High Level Design (5-10 pages): Describing implementation aspects/details.
- Semester B:
  - Development – Implementing the solution.
  - Midterm (5.6-24.6) – Presenting the project (in English) for a an external examiner.
- Summer Semester:
  - Project Completion and Final.
  - Project Exhibition event (8.9.2026) – The course is presented (in hebrew) to external judges.

Important Notes

- Teams size: 2-3 students
- Involvement of all students in the team in the various aspects of the project is required.
- Fulfillment of obligations (meeting; deadlines; submitting documents, etc.)
- Grade can vary from student to student.
- Teams must work with GITHUB to store the code, documents, and other workshop products.
- The repository must be public (unless there is an extreme case that justifies otherwise).
- Regular bi-weekly Team meetings.

Grade Composition

- 20% Midterm – given by the extrenal examiner during the project presentation (in English)
- 10% Project scoring in Project Exhibition ranking.
- 70% - Remainder
  - **[Team] Project Content:** Mainly Quality & Technical depth. (25%)
    - Innovation and usability are mostly evaluated by the external supervisor and judges.
  - **[Team] Project Management:** Punctuality, Meeting Regularity and reporting (25%)
  - **[Individual] Team work:** Ownership, Meeting Attendance, Progress update and Knowledge sharing (20%)

Bi-Weekly Sprint Meeting **&** Communication

- Purpose: Track progress and Planning.
- Length: 1 hour.
- Each team member update on:
  - Status update (Progress, blockers, achievements)
  - Next sprint plan (next steps , goals, tasks)
  - Decision log (design choices, trade-offs)

- One of the team memebers is the sprint master.
- His responsibility is to create a meeting summary documenting the progress and involvement of all participants.
- Each group should open a communication channel (preferably a slack channel).

Projects Artifacts (Semester A)

Project Proposal

- **Purpose**: Demonstrate problem understanding and scoping.
- **Includes**:
  - Problem definition + motivation
  - Value proposition (why important)
  - Literature / related work
  - Proposed solution & architecture concept
  - Success metrics
  - Timeline & milestones

Technical Spec

- **Purpose**: Translate the idea into an actionable engineering plan – design an architecture.
- **Includes**:
  - System architecture diagram (components + interactions)
  - Data flows (input/output schemas, storage plan)
  - APIs & tools used
  - Risk assessment (technical + operational)
  - Task breakdown (who owns what)

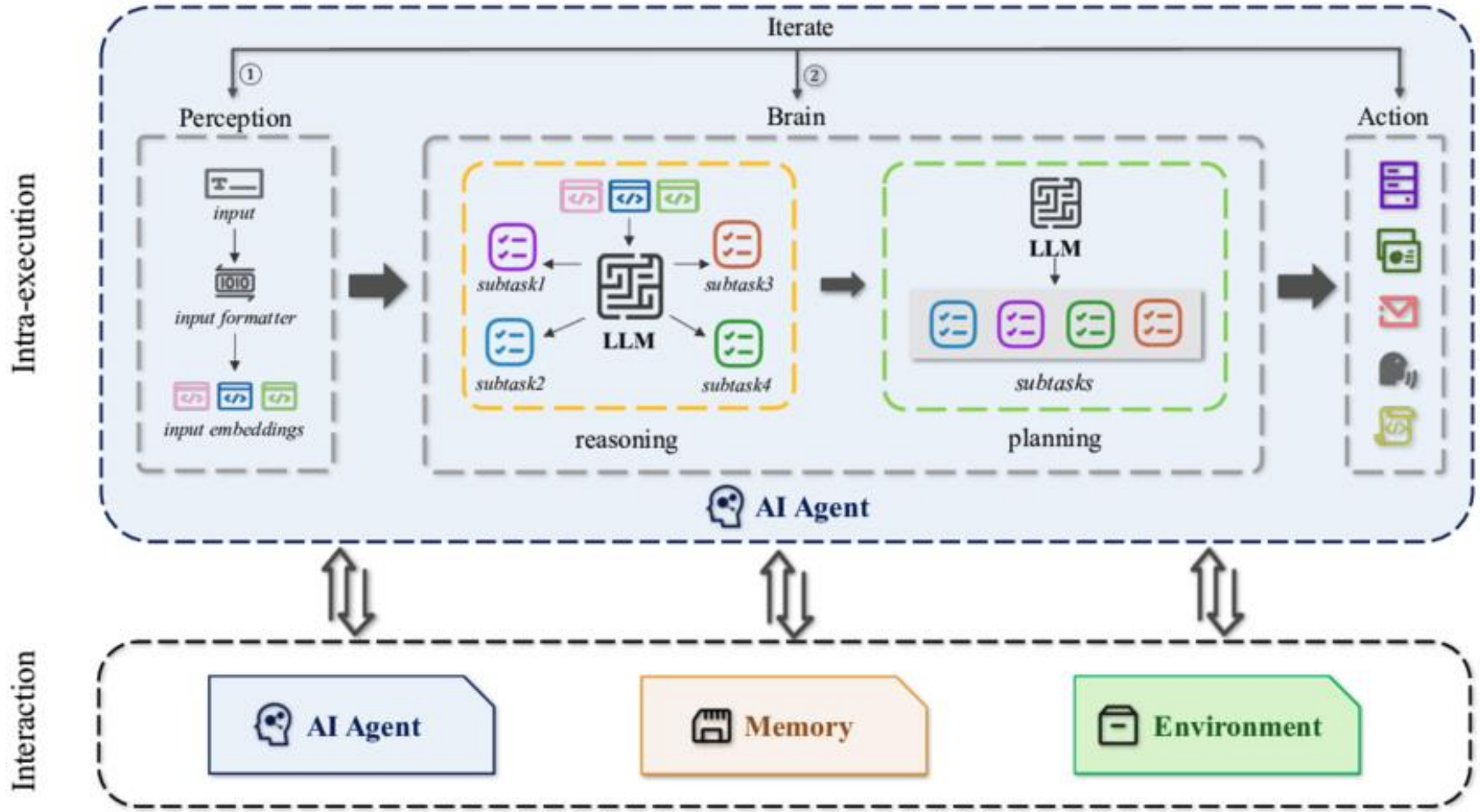
AI Agents

A short intro

Traditional software vs. Agentic Apps

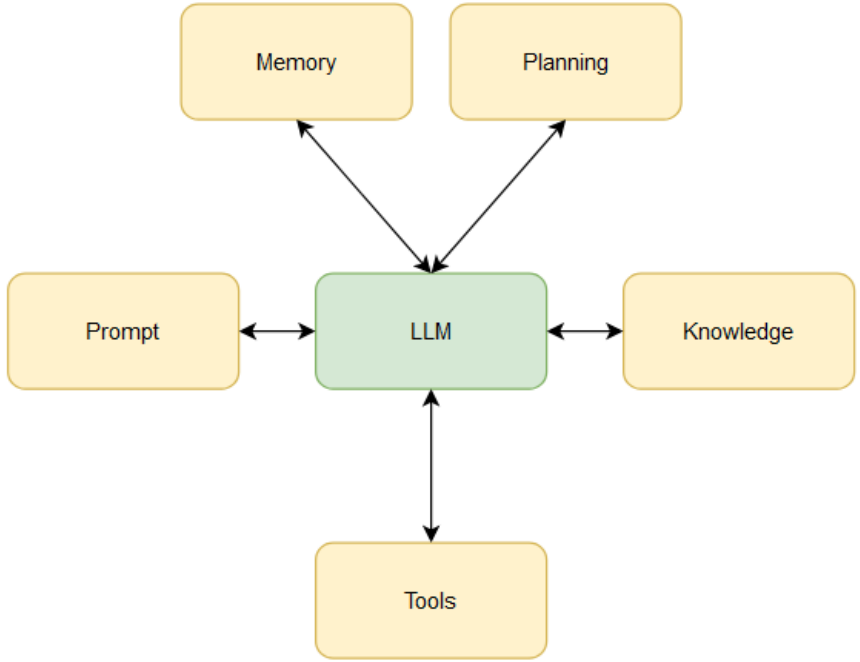
- **Traditional Software Applications**
  - Operate on **structured, unambiguous data**.
  - Use **deterministic, human-defined control logic** — the developer decides every rule and flow.
  - Behavior is **predictable** and repeatable.
- **Agentic AI Applications**
  - Consume **unstructured, multimodal, and ambiguous data**.
  - **Generate control logic dynamically** using LLM reasoning and planning.
- The boundary between **data and control logic is collapsing** — data shapes behavior in real time.
- Designing, testing, and governing AI agents requires **new paradigms grounded in statistical reasoning**, not certainty.

What is an AI agent?



Agent Main Components

- **Prompts** are instructions that give information to LLM about its objective, behavior and plan. The agent's performance is very dependent on the quality of a prompt.
- **Memory**: Agents complete a complex task by first breaking down into sub-tasks than executing tools to finish sub-tasks. For this, the model needs to remember its previous steps.
- **Knowledge:** Without the knowledge of the field, agent can not solve or even understand the task. So either the LLM must be fine-tuned to have the knowledge or we can create a tool to extract the knowledge from a database.
- **Planning:** Complex problems often need a chain-of-thought approaches. Agents forms a plan by using a combination of two methods:
  - Task and question decomposition: Breaking down the task or question into smaller parts
  - Reflection or critic: Frameworks such as React are used to critic the plan generated by the agent.
- **Tools:** Executable functions, APIs or other services that allow agents to complete their duties.



Project Examples

The 3 categories

Category 1– Applications

- **Personal Assistant (RAG + agent planning)**
  - Agents that performs deep research on a personal data or team data (slack)
  - Finance helper that generate recommendations and action related to billing, investments, etc.
- **Negotiation Agent (multi-goal reasoning)**
  - Agent interacts with humans or another agent to negotiate pricing, scheduling, etc.
  - Evaluates utility, trade-offs, and simulates persuasion strategies.
- **SRE agents: Server Monitoring and Remediation Agent**
  - Periodically monitors server via iPerf3 / Prometheus.
  - Suggests or *executes* actions (restart container, free memory).
- **Building MCP to a specific common system**
  - E.g. MAMA, trade platform, etc.
- **Code/Development assistent agent.**
  - Automatic scenario generation to specific agents.
  - Automatic execution of benchmarks.
  - Observability and optimization.

Category 2: Infrastructure / Framework – *Build something that others can use*

- **Multi-agent collaboration framework**
  - A runtime where agents can assign tasks to other agents and merge results.
  - Includes:
    - Shared memory (vector DB)
    - Messaging router
  - Agent-to-agent communication patterns
- **Mini Agenthub**
  - Build a micro-platform where developers can deploy simple agents via API.
  - Supports:
    - Register agent
    - Trigger agent task
    - Track results
  - **I.e., Using** FastAPI + Docker + LangGraph
- Continuous learning abilities
- **MCP (Model Context Protocol) tool store**
  - Build a new MCP tool store – general or specialized: search, email summarization, or PDF extraction.
  - Demonstrates:
    - Context sharing
    - State interoperability
    - Tool lifecycle
  - **Output:** An MCP tool others can use in ChatGPT / agent frameworks.
- Building novel Agent Component
  - Contribute agents' components to existing open-source platforms.
  - E.g., New observability/reliability mechanisms.
- Make using MCP easier. More safe

Category 3: Research – *Study the behavior of agents and LLMs*

- **Safety-aware Agent: Policy Compliance + Red Teaming**
  - Agent receives a policy and must act within the boundaries.
  - Attack scripts and measure how often the agent violates policy.
- How does prompting or reasoning depth affect compliance?
- Which agent architecture is more robust (reflection / ReAct / plan-and-execute)?
- The effect of Agents in Gaming/Game experience.
- **Reasoning Depth vs. Performance Benchmark**
  - Use different agent reasoning mechanisms: Direct answer, Chain-of-thought/ deliberate reasoning, ReAct or RePlan
  - And compare: Quality / Hallucination rate / Policy violations
- **Emotional Companion / Story Agent**
  - Agent evolves personality based on conversation history.
  - Study the react of people to agents on specific domain.

Resources

- https://www.mlsysbook.ai/contents/core/introduction/introduction.html
- https://research.ibm.com/blog/what-are-ai-agents-llm
- https://medium.com/@aydinKerem/what-is-an-llm-agent-and-how-does-it-work-1d4d9e4381ca
- https://medium.com/@aydinKerem/ai-agents-design-patterns-explained-b3ac0433c915