Multimodal NLP and Agentic AI

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

LMH
Lady Margaret Hall

July 4, 2025

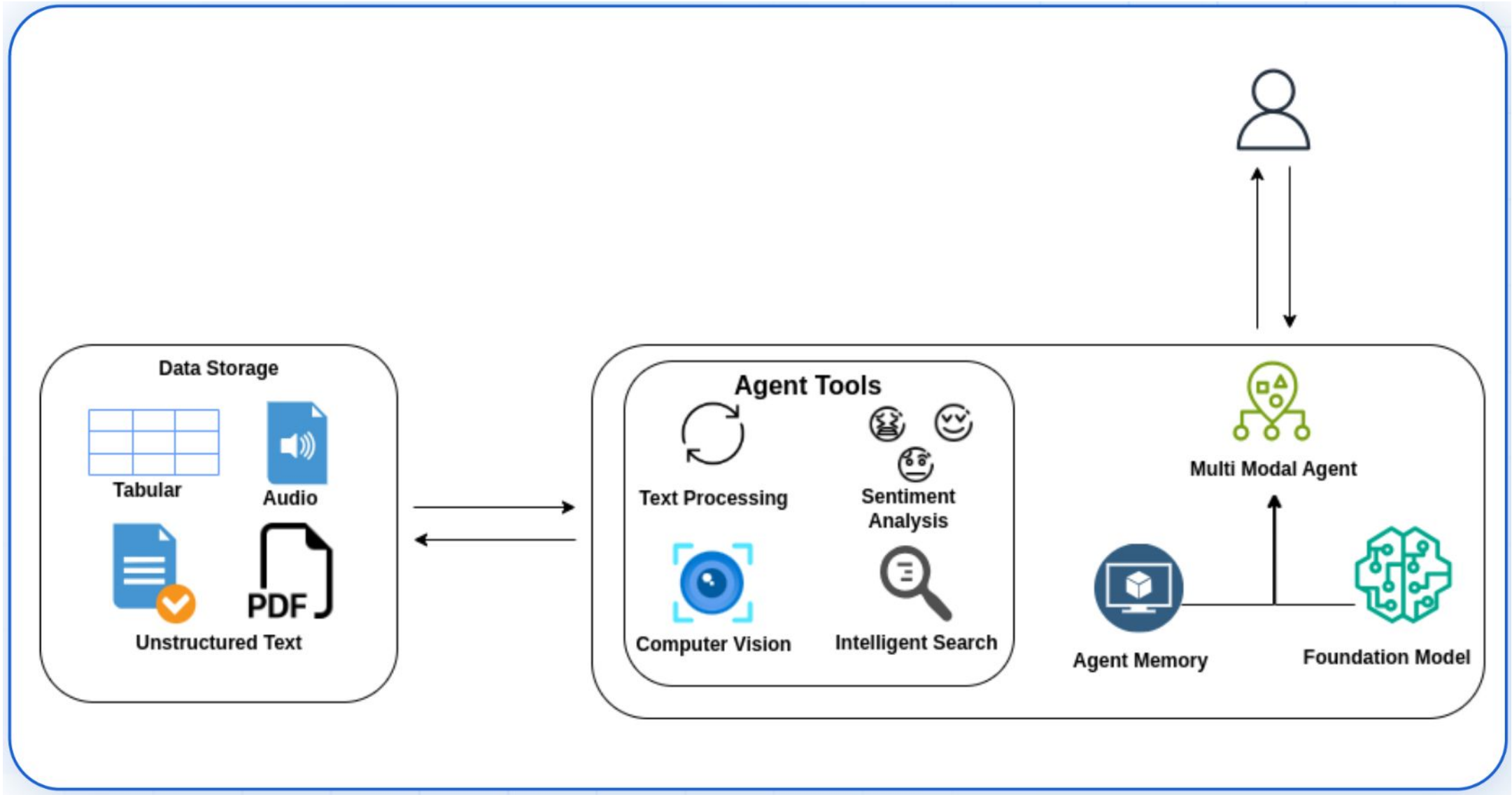

Data Storage
Tabular
Audio
PDF
Unstructured Text
Agent Tools
Text Processing
Sentiment Analysis
Computer Vision
Intelligent Search
Multi Modal Agent
Agent Memory
Foundation Model

Table of Contents

1. Motivation
2. Learning Outcomes
3. Vision-Language Models
    1. CLIP: Contrastive Language-Image Pretraining
    2. VisualBERT
    3. FLAVA: Fusion of Language And Vision Architecture
4. Multimodal Architectures
    1. Categories of Multimodal Architectures
    2. Challenges in Multimodal Integration
5. Agentic AI
    1. What is Agentic AI?
    2. Reactive vs. Proactive Agents

Table of Contents

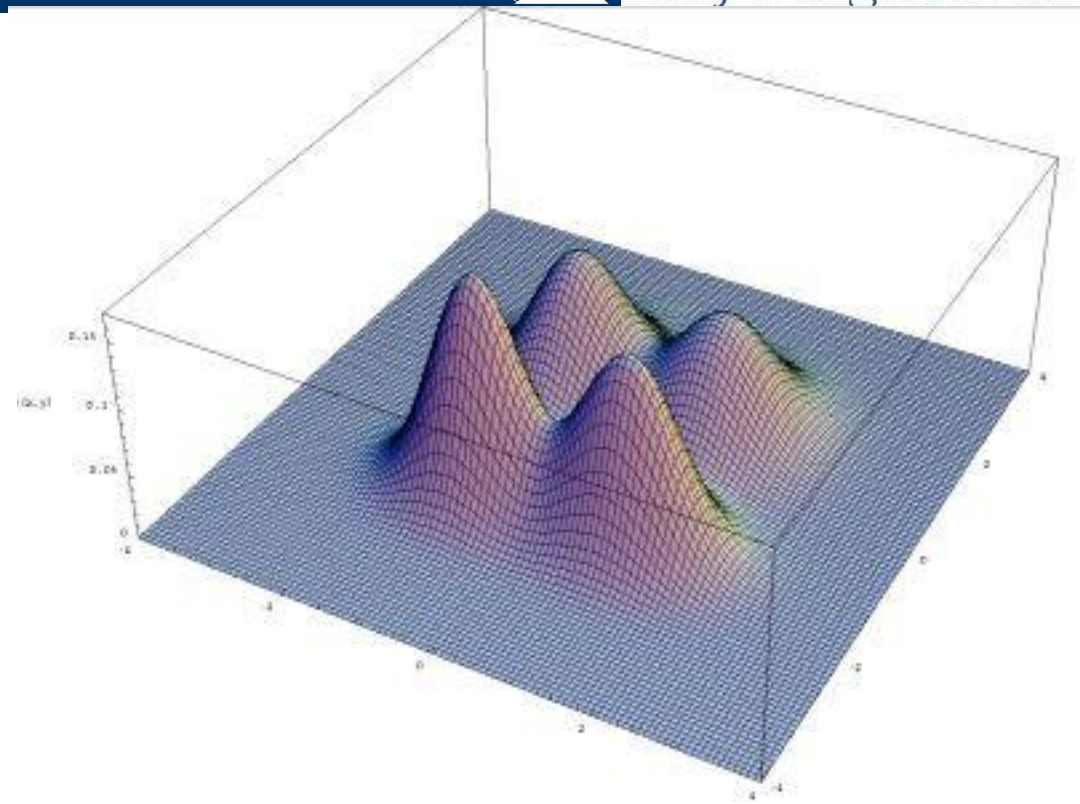
3. Agent Loop

6. Open-Source Agentic Frameworks
   1. Auto-GPT
   2. BabyAGI
   3. Other Notable Frameworks
7. Continual Learning in Agents
   1. Why Continual Learning?
   2. Challenges in Continual Learning
8. Safety, Ethics Control
   1. Safety in Agentic Systems
   2. Ethics and Control
9. Future Directions

Motivation

- Traditional NLP models are **unimodal** (text-only) and **reactive**.
- Modern AI agents need to perceive, reason, and act in complex environments using **multiple modalities** (e.g., vision, text, speech).
- **Multimodal systems** and **agentic AI** aim to build goal-directed, autonomous, and continuously learning agents.
- Rise of agent frameworks (Auto-GPT, BabyAGI) and vision-language models enables new research frontiers in decision-making, robotics, and assistive AI.

What is multimodality?



multimodal adjective

mul·ti·mod·al ˌməl-tē-ˈmō-dᵊl -ˌtī-

: having or involving several modes, modalities, or maxima

multimodal distributions

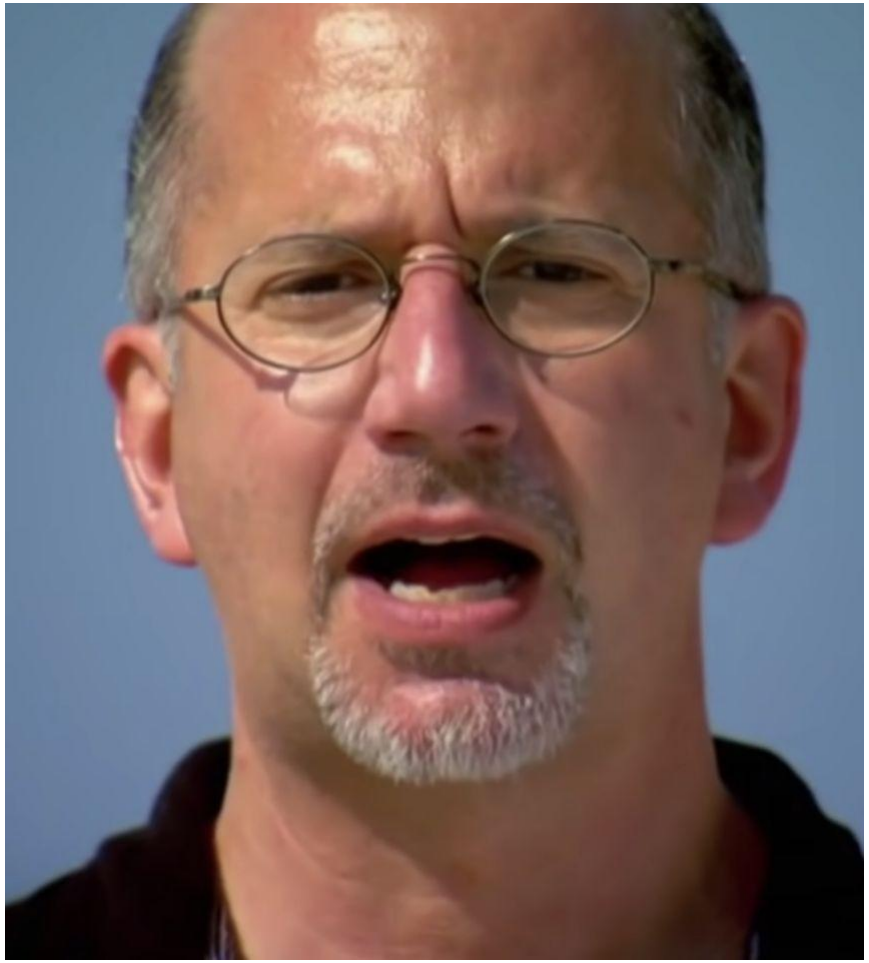
multimodal therapy

In our case, focusing on NLP: text + one or more other *modality* (images, speech, audio, olfaction, others). We'll mostly focus on images as the other modality.

Multimodal brains

McGurk effect (McGurk & MacDonald, 1976)

https://www.youtube.com/watch?v=2k8fHR9jKVM



Learning Outcomes

- Explain core vision-language models (e.g., CLIP, VisualBERT, FLAVA).
- Compare multimodal architectures and their integration mechanisms.
- Understand the principles of agentic AI and agentic loops.
- Differentiate between reactive and proactive agent behavior.
- Describe continual learning challenges in agentic systems.
- Critically evaluate frameworks like Auto-GPT and BabyAGI.
- Reflect on safety, control, and ethical issues in agentic AI.

Vision-Language Models

CLIP: Contrastive Language-Image Pretraining

- Developed by OpenAI (Radford et al., 2021)
- Trained on 400M (image, text) pairs
- Learns a joint embedding space using contrastive loss
- Image and text encoders are trained to match correct pairs

Use cases:

- Zero-shot classification
- Semantic similarity
- Prompt-based image querying

CLIP: Contrastive Language-Image Pretraining

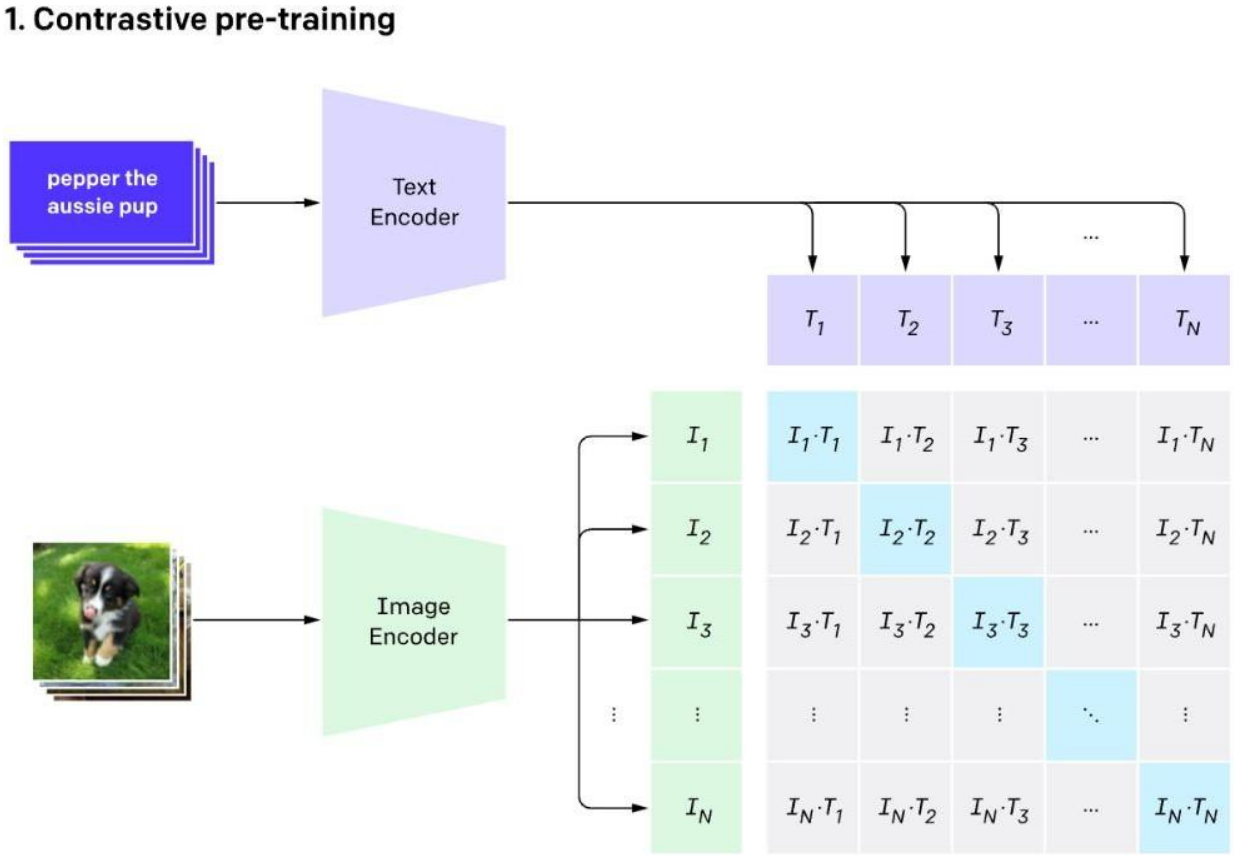
Architecture:

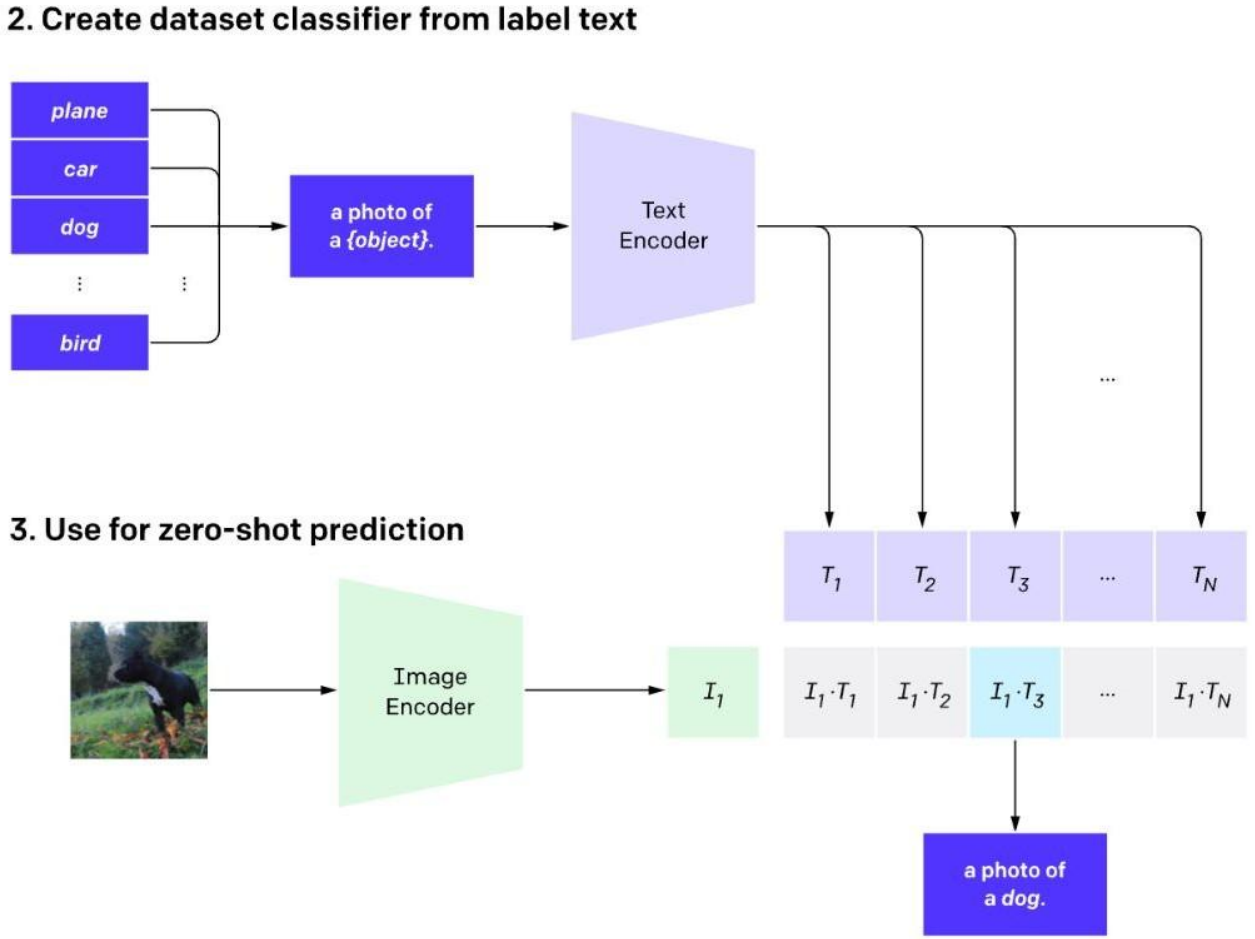
- ResNet / ViT for images
- Transformer for text
- Cosine similarity loss

Paper: CLIP: Learning Transferable Visual Models From Natural Language Supervision https://arxiv.org/abs/2103.00020

CLIP (Radford et al. 2021)

Exact same contrastive loss as earlier, but.. Transformers and *web data*!

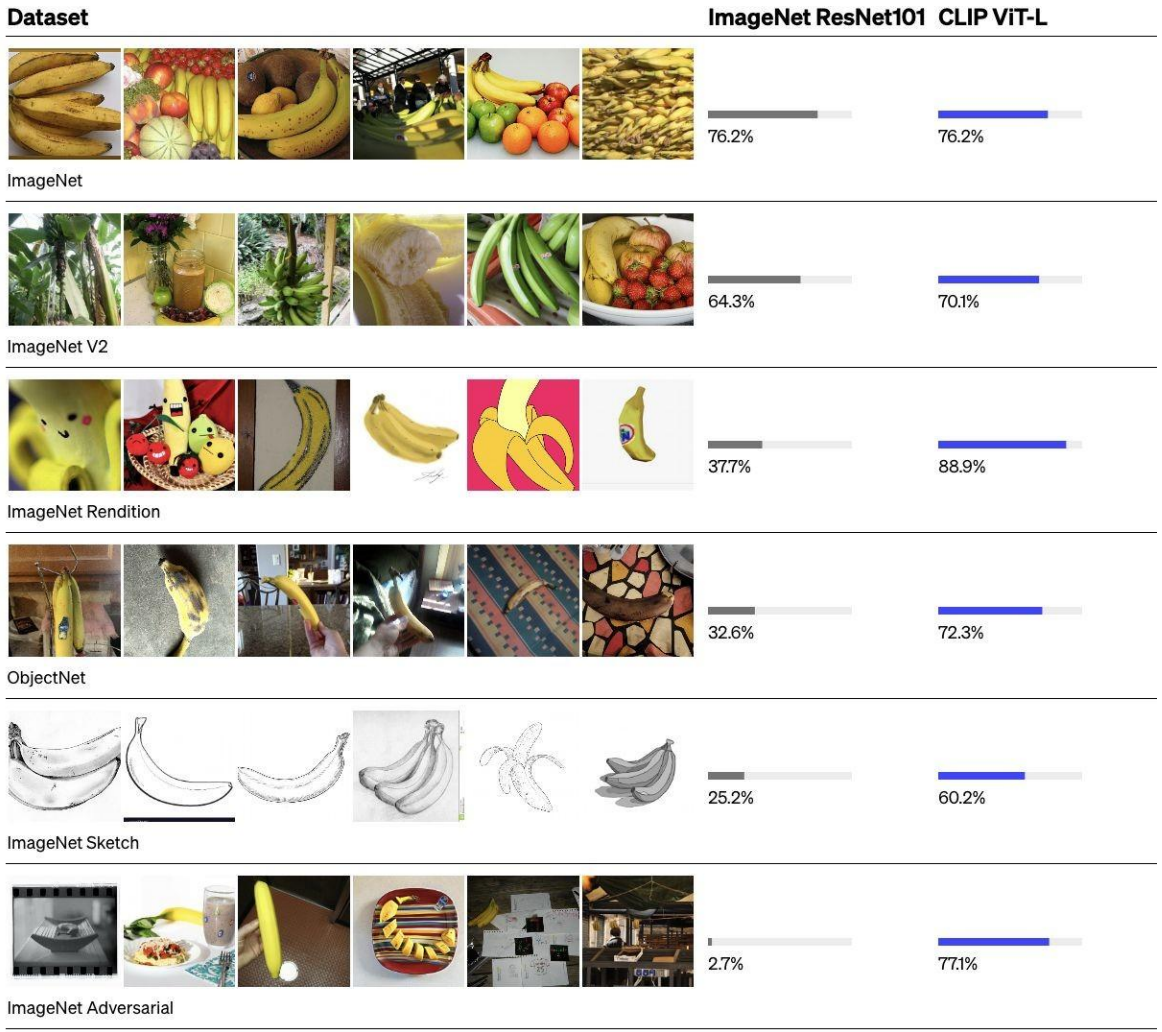




CLIP Robustness

IMHO one of the best papers ever written in our field: extremely thorough, worth a close read.

Generalizes MUCH better →



VisualBERT

- Joint Transformer-based model: processes both image regions and text.
- **Early fusion:** image embeddings are added as input tokens to the Transformer.
- **Pretraining objectives:**
  - Masked Language Modeling (MLM)
  - Next Sentence Prediction (NSP)
  - Image-Text Alignment

Applications:

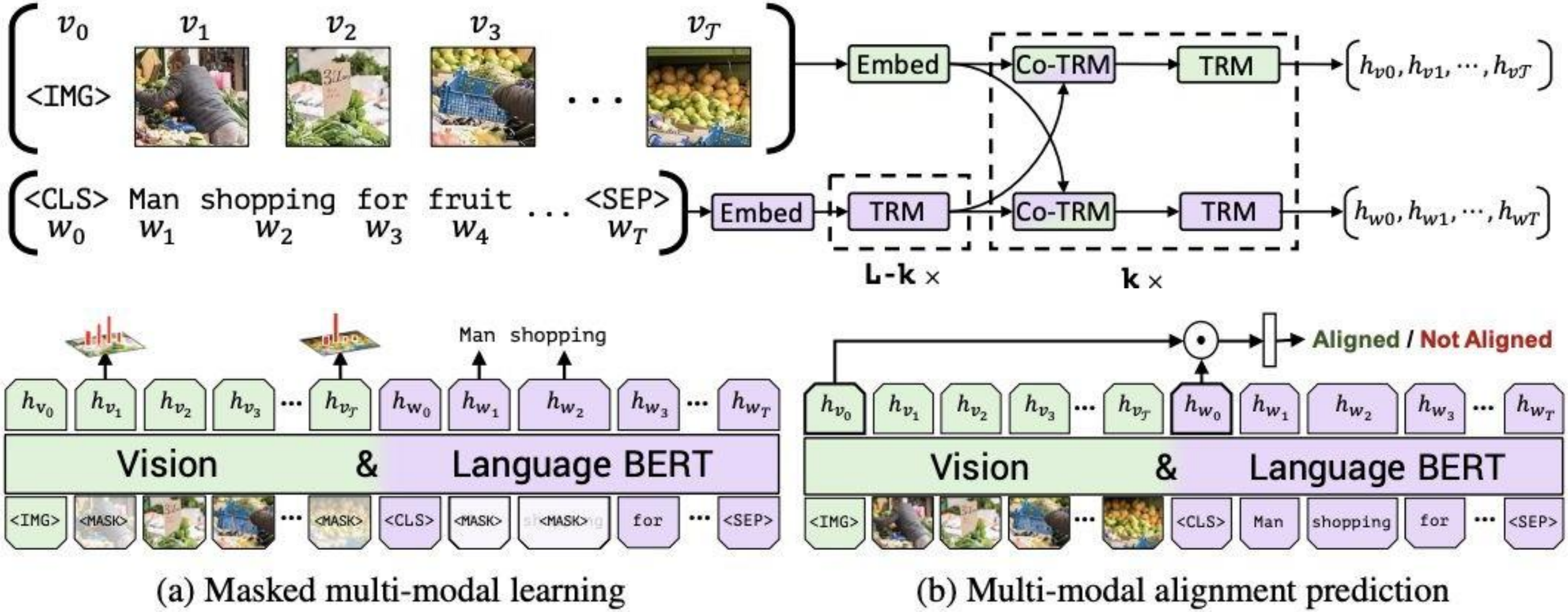
- Visual Question Answering (VQA)
- Image captioning
- Visual commonsense reasoning

Paper: VisualBERT: A Simple and Performant Baseline for Vision and Language

Visual BERTs: VisualBERT

VisualBERT Li et al. 2019

Visual BERTs: ViLBERT



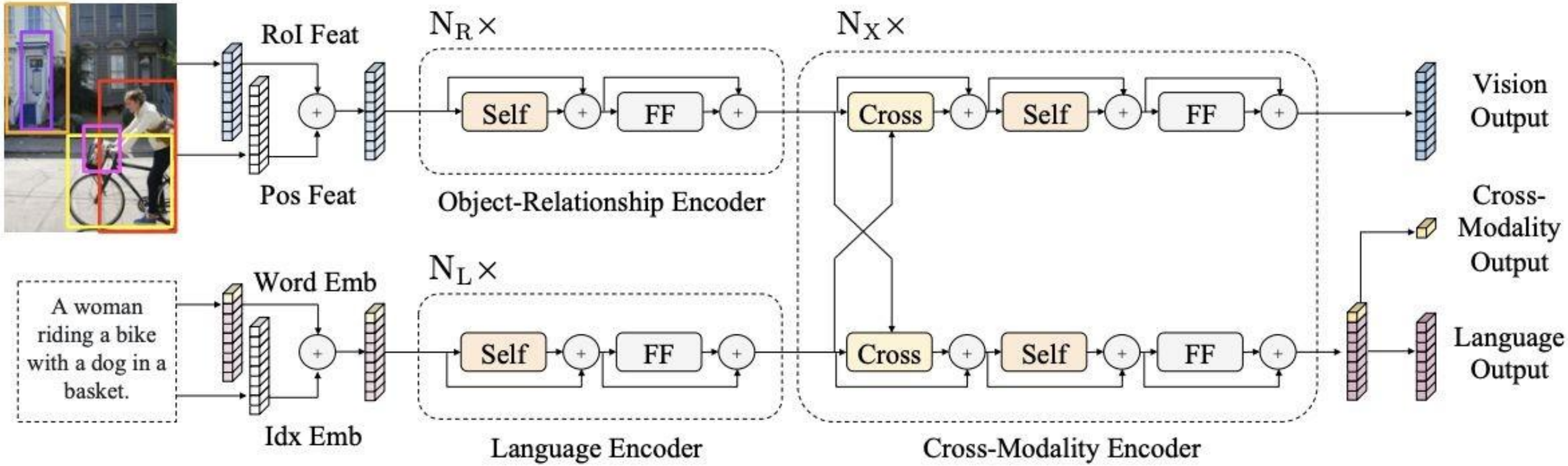
(a) Masked multi-modal learning

(b) Multi-modal alignment prediction

ViLBERT Lu et al. 2019

Visual BERTs: LXMERT

Learning Cross-Modality Encoder Representations from Transformers



LXMERT Tan & Bansal 2019

Visual BERTs: Supervised Multimodal Bitransformers

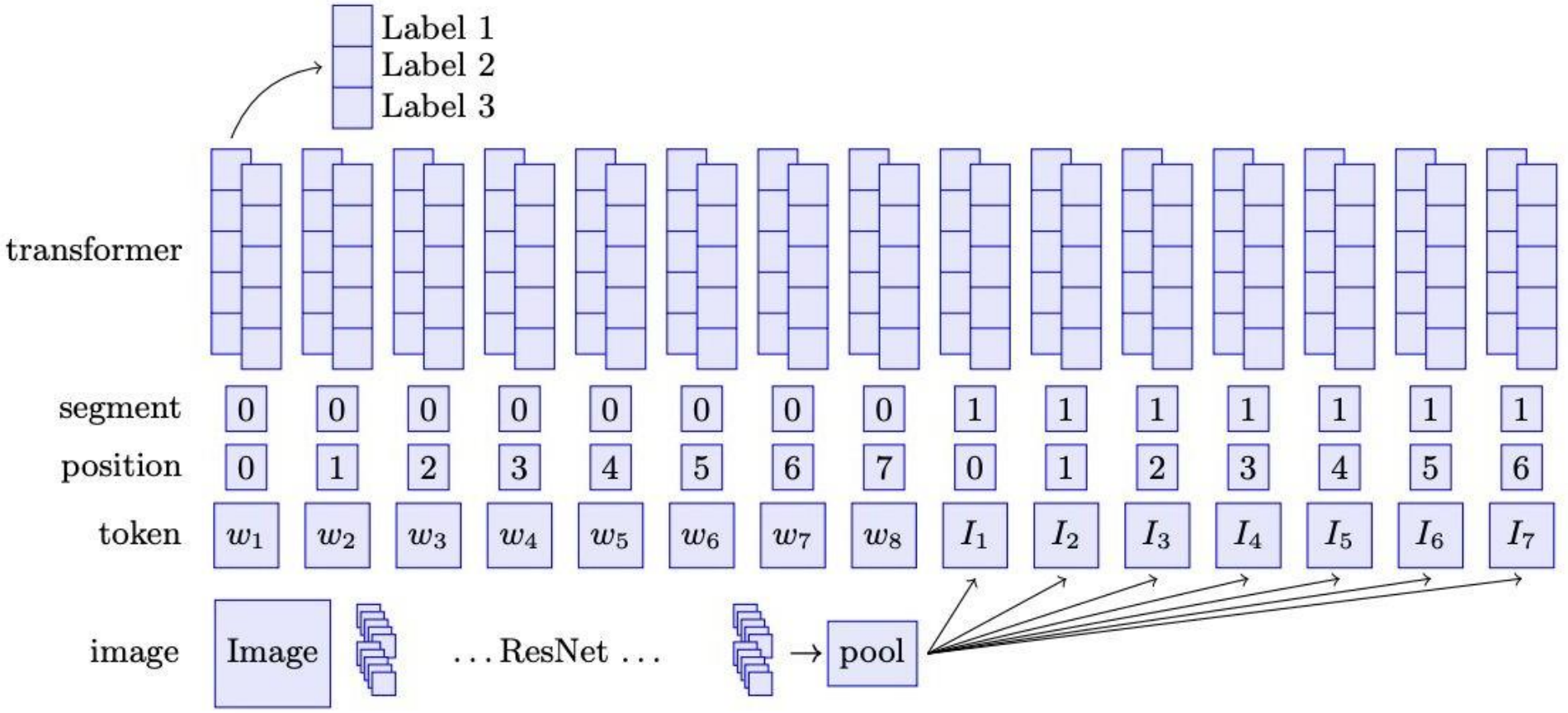
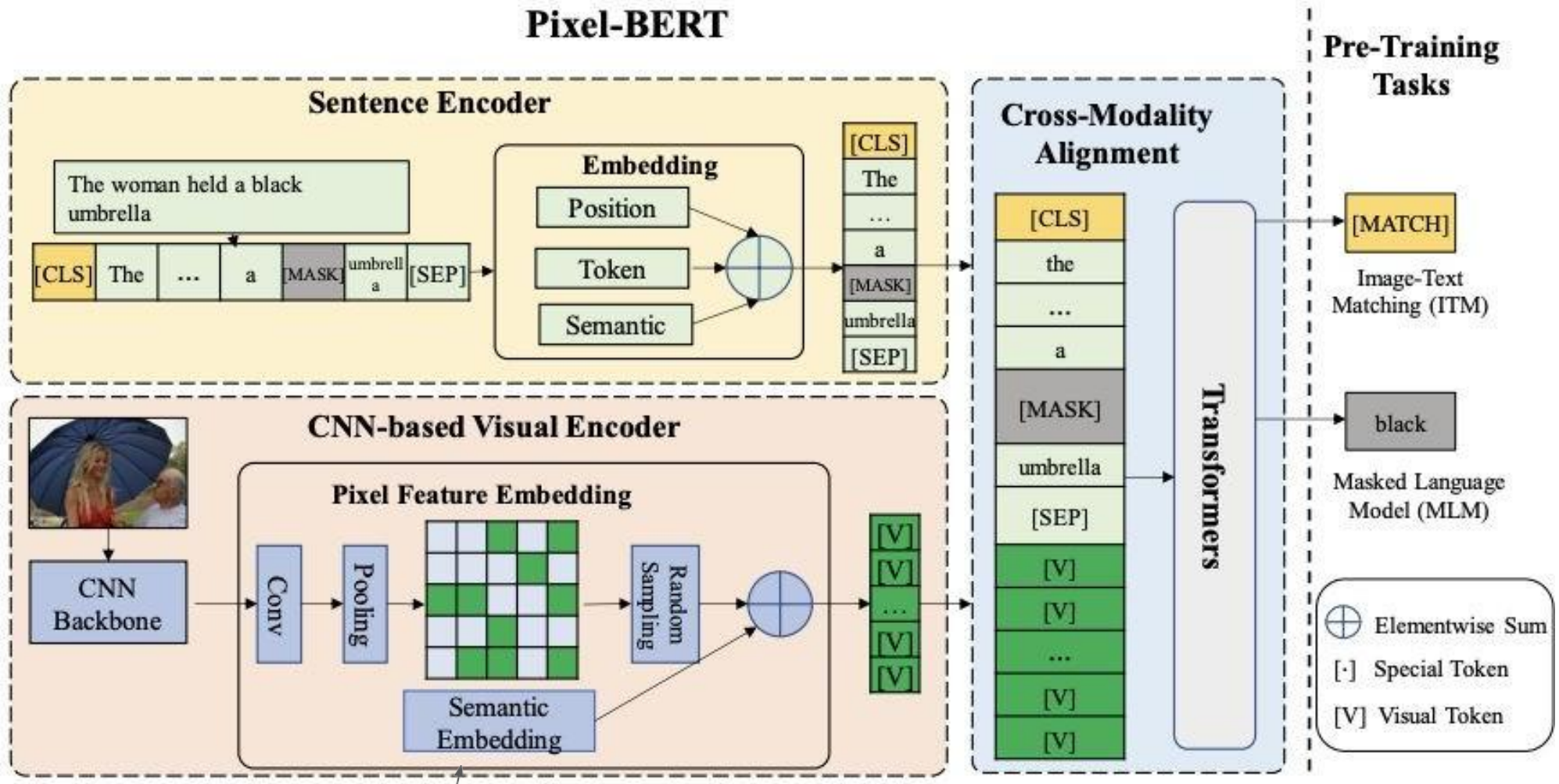


Figure 1: Illustration of the multimodal bitransformer architecture.

MMBT Kiela et al. 2019

Visual BERTs: PixelBert



Misnomer: they mean segment embedding

PixelBert Huang et al. 2020

FLAVA (Facebook AI)

- FLAVA = **Fusion of Language And Vision Architecture**
- Handles both unimodal and multimodal tasks
- **Three towers:** vision, language, multimodal
- Self-supervised objectives: masked modeling, contrastive learning

Capabilities:

- Text classification
- Image classification
- Visual Question Answering (VQA)
- Image captioning

Paper: FLAVA: A Foundational Language And Vision Alignment Model

FLAVA (Singh et al., 2021)

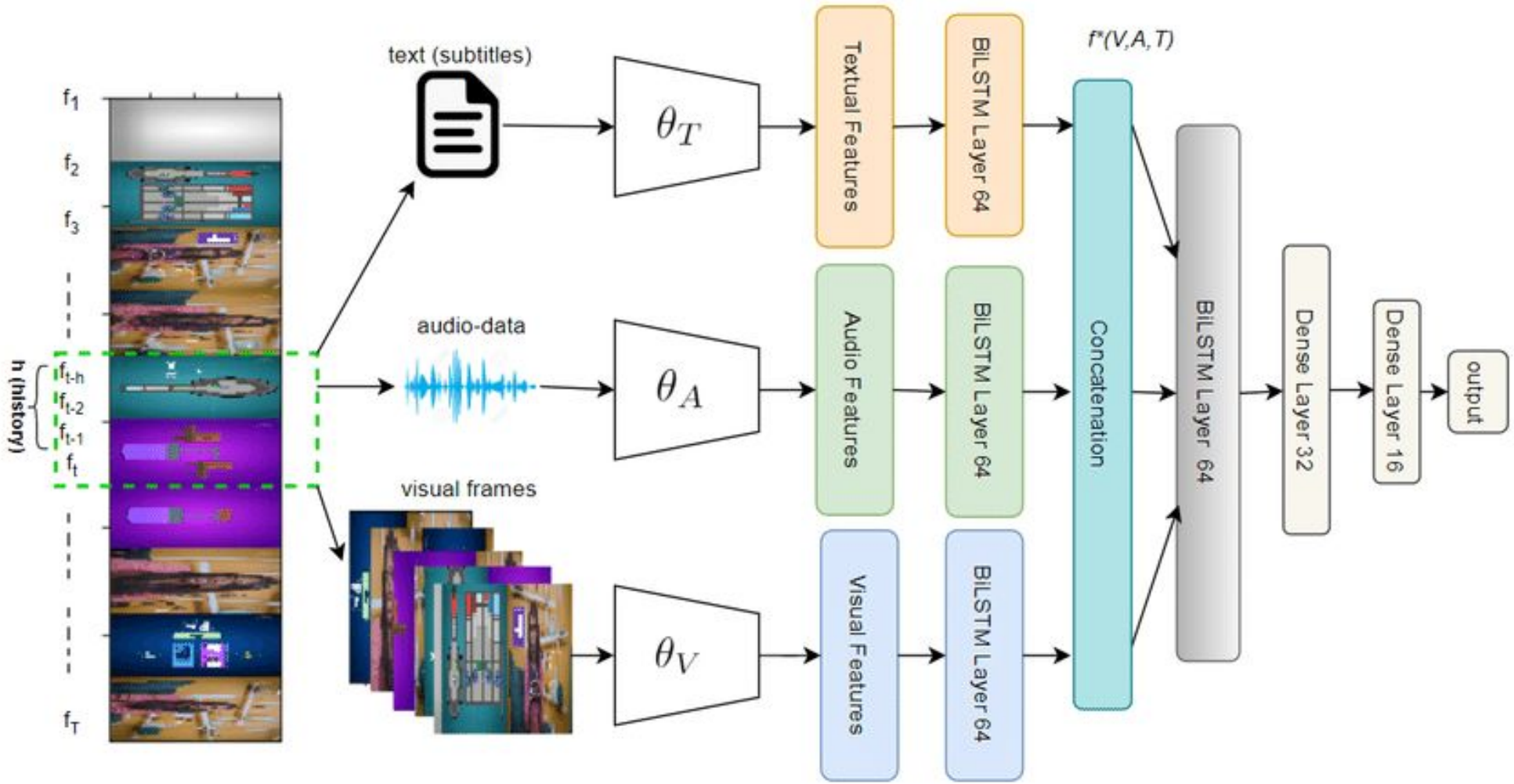
Holistic approach to multimodality.
One foundation model spanning V&L, CV and

NLP. Jointly pre-trained on:

- unimodal text data (CCNews + BookCorpus)
- unimodal image data (ImageNet)
- public paired image-text data (70M)

All data/models are publicly released.

Multimodal Architectures


text (subtitles)
audio-data
visual frames
f_1
f_2
f_3
f_{t-h}
f_{t-2}
f_{t-1}
f_t
f_T
h (history)
θ_T
θ_A
θ_V
Textual Features
Audio Features
Visual Features
BiLSTM Layer 64
BiLSTM Layer 64
BiLSTM Layer 64
f*(V,A,T)
Concatenation
BiLSTM Layer 64
Dense Layer 32
Dense Layer 16
output

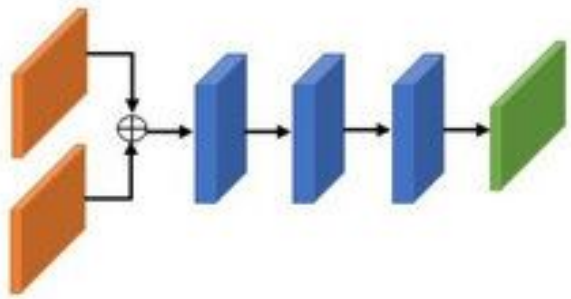
Categories of Multimodal Architectures

- **Early Fusion:** Combine raw modalities at the input level.
  Example: VisualBERT adds image embeddings as tokens to the Transformer.
- **Late Fusion:** Process each modality separately, then merge representations.
  Example: CLIP encodes images and text independently, then compares embeddings.
- **Hybrid / Cross-modal Attention:** Enable interactions between modalities at multiple stages using attention mechanisms.

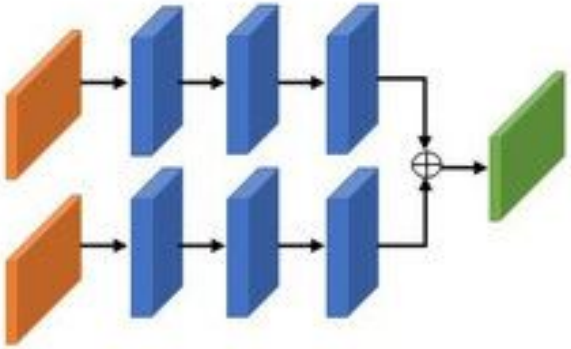
Design Considerations:

- **Alignment:** How to align representations across modalities.
- **Modality Dropout:** Handling missing or noisy modalities during training or inference.
- **Shared vs. Modality-Specific Layers:** Deciding which layers are shared and which are unique to each modality.

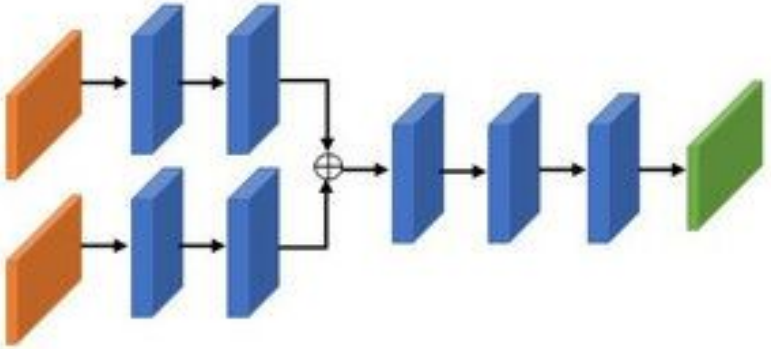
Categories of Multimodal Architectures



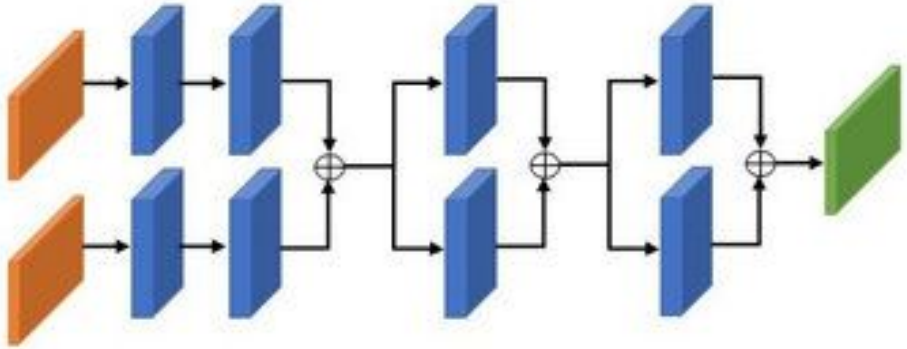
(a) Early Fusion



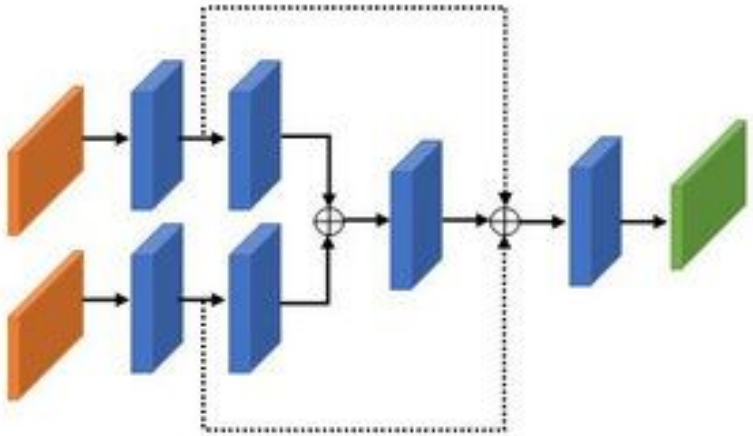
(b) Late Fusion



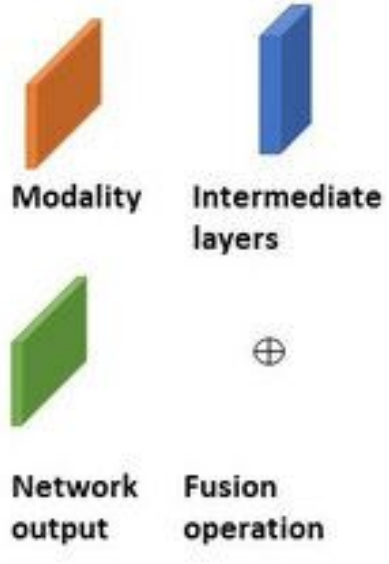
(c) Middle Fusion - fusion in one layer



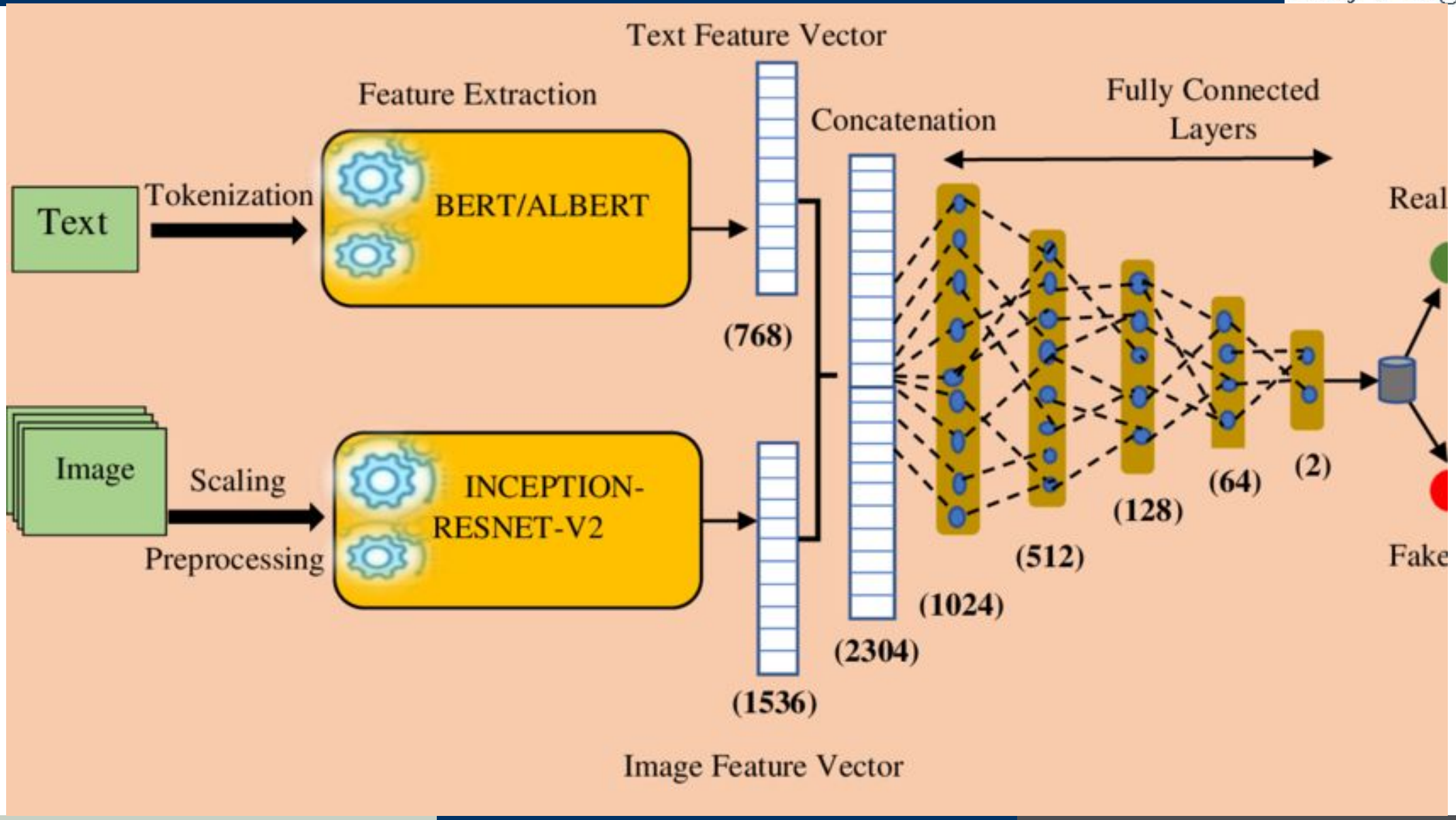
(d) Middle Fusion - deep fusion



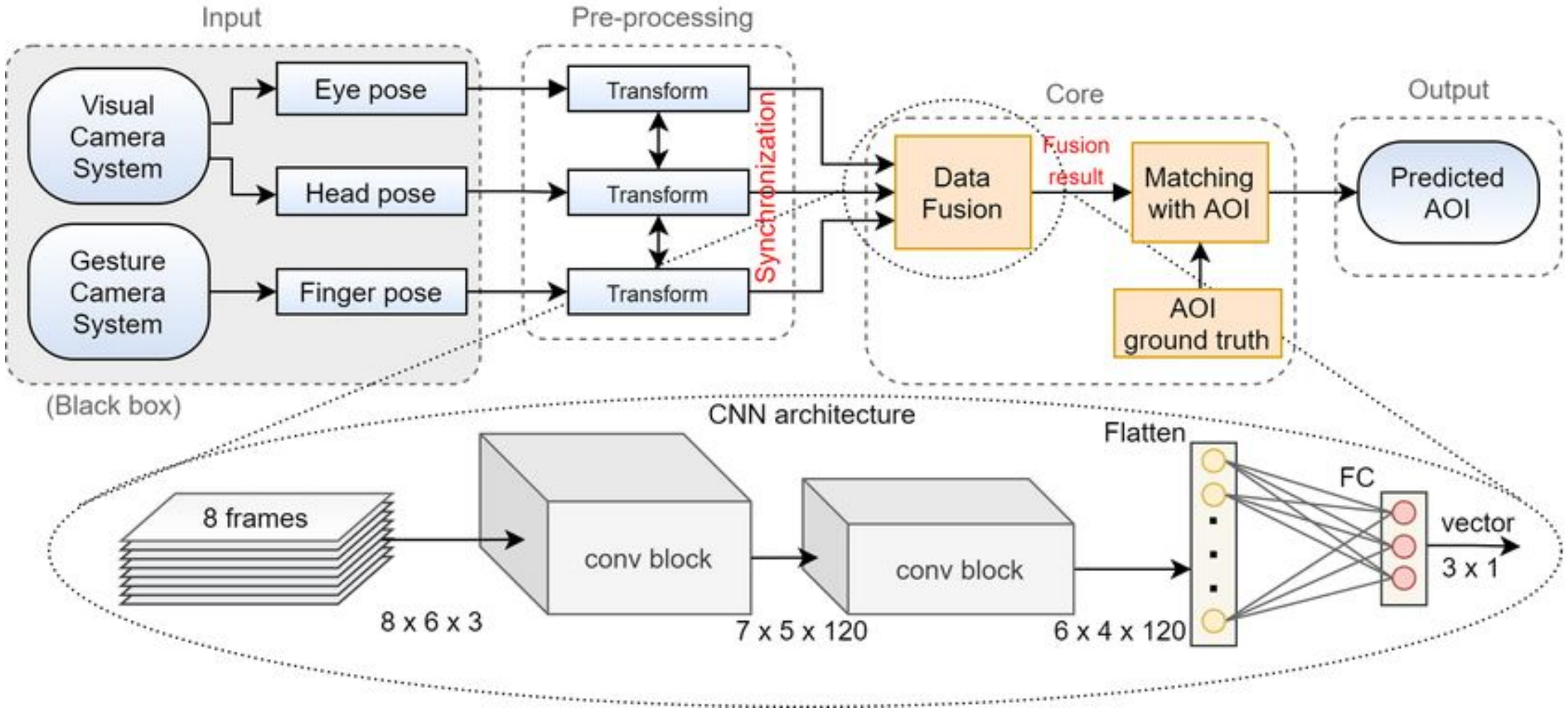
(e) Middle Fusion – short-cut fusion



Early Fusion



Late Fusion


Input
Pre-processing
Visual Camera System
Gesture Camera System
Eye pose
Head pose
Finger pose
Transform
Transform
Transform
Synchronization
Core
Data Fusion
Fusion result
Matching with AOI
AOI ground truth
Output
Predicted AOI
(Black box)
CNN architecture
8 frames
8 x 6 x 3
conv block
7 x 5 x 120
conv block
6 x 4 x 120
Flatten
FC
vector
3 x 1

Challenges in Multimodal Integration

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

MultiModal Learning Challenges

Representation

Finding a good representation is crucial for the success of multimodal models that combine different modalities such as image, text, and audio.

Fusion

Fusing information from different modalities to perform a prediction task is the core of multimodal learning and can be challenging due to the diverse nature of multimodal data.

Alignment

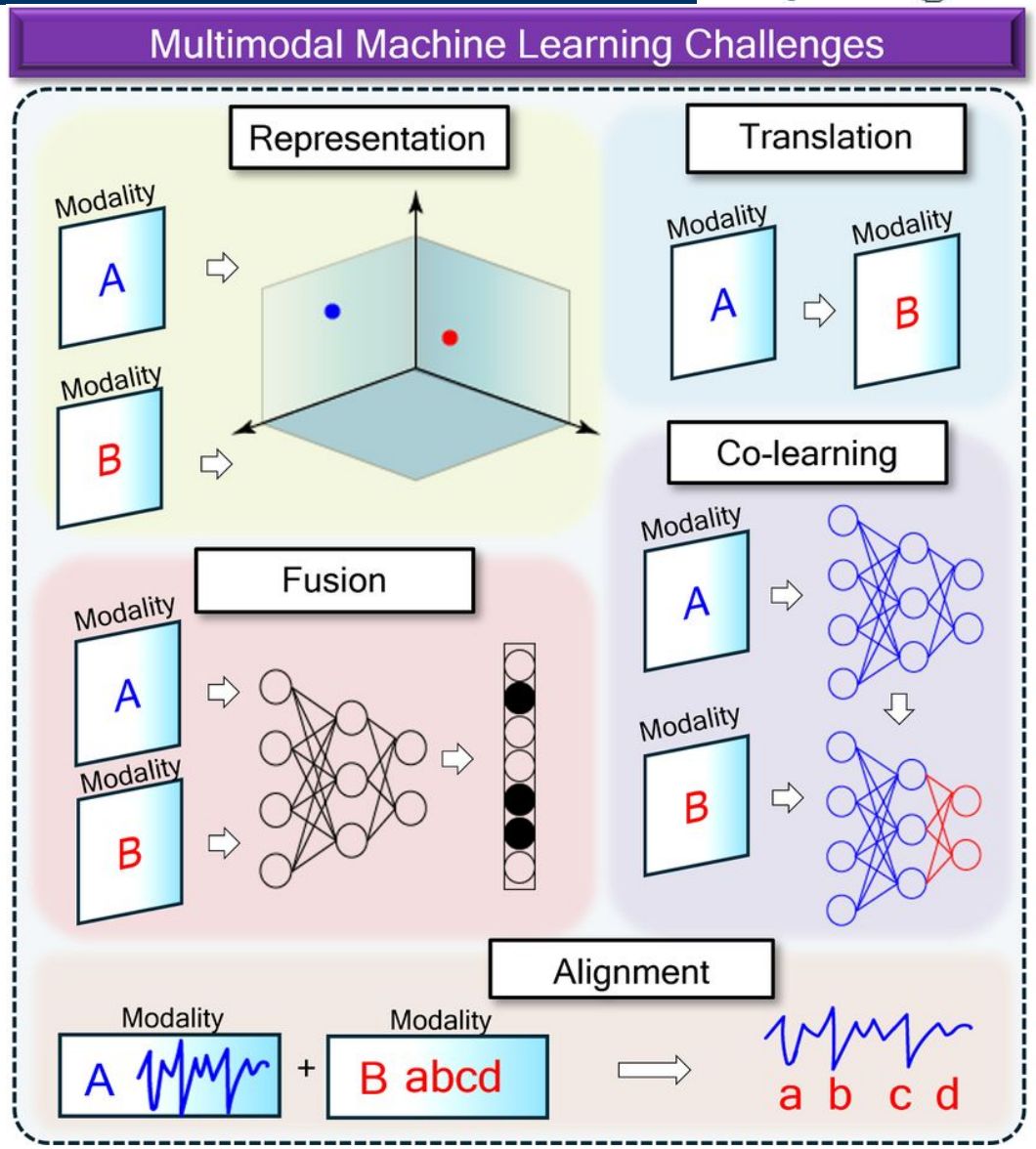
Aligning data from different modalities to create modality-invariant representations is necessary to combine information effectively.

Translation

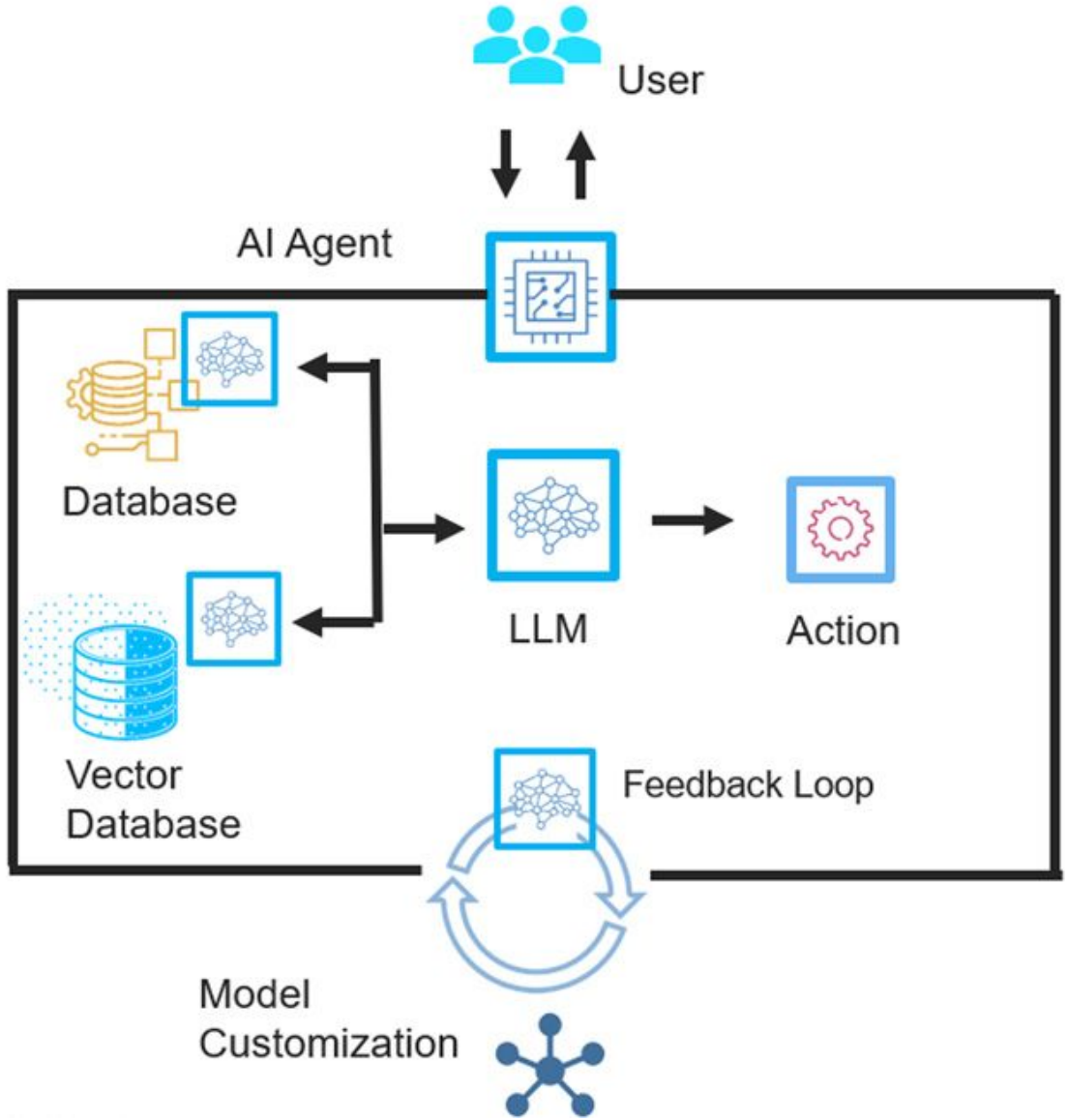
Translating data from one modality to another is a crucial task in multimodal learning, but can be difficult due to the subjective and open-ended nature of translations.

Co-learning

Co-learning aims to transfer knowledge learned through one or more modalities to tasks involving another. It is useful in medical diagnosis combining CT and MRI scans.



Agentic AI


User
AI Agent
Database
LLM
Action
Vector
Database
Feedback Loop
Model
Customization

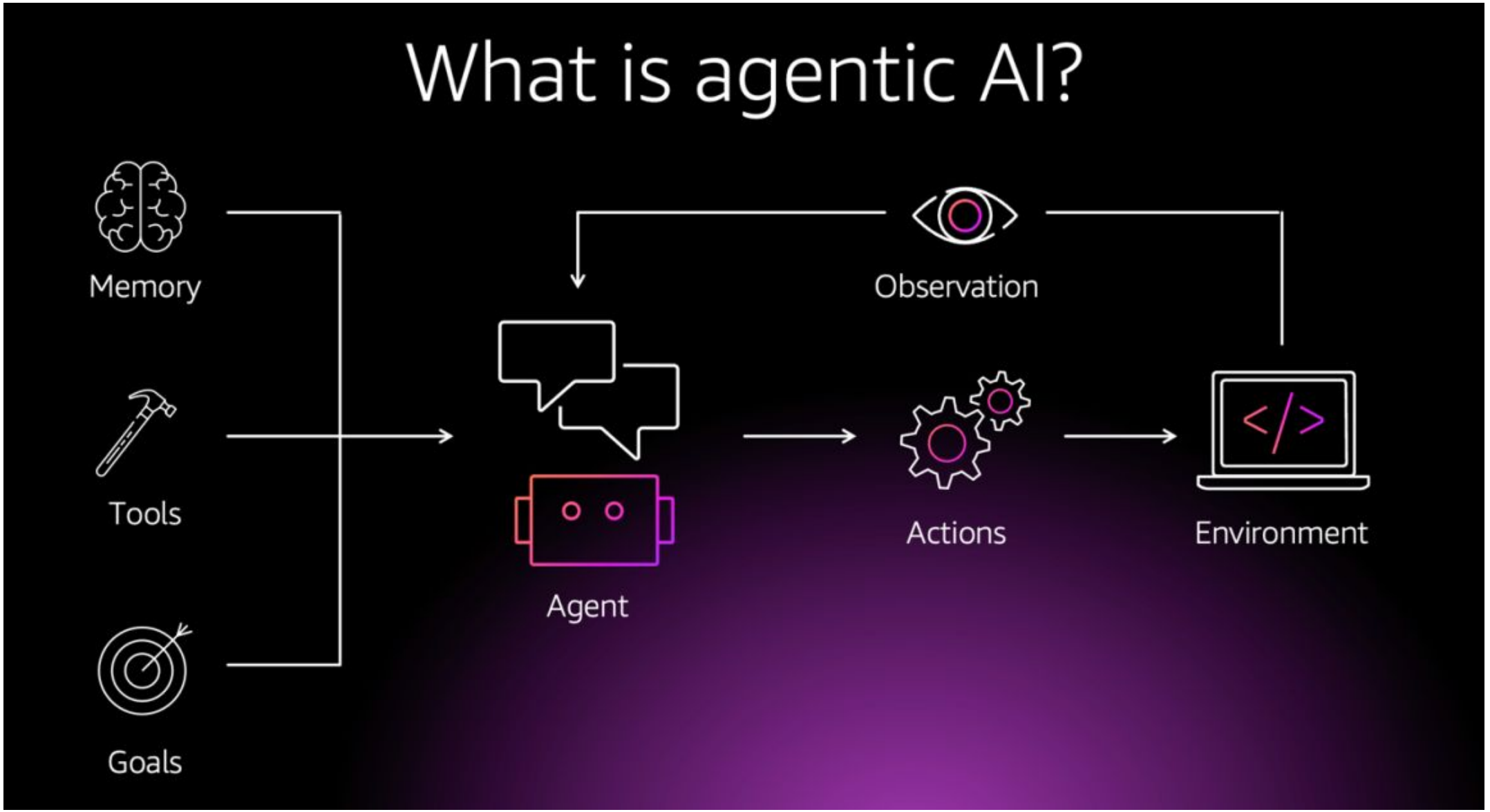
What is Agentic AI?

- AI systems that **perceive**, **reason**, **plan**, and **act** to fulfill goals.
- **Agent loop:** Observe → Plan → Act → Reflect → Learn
- Inspired by cognitive science, robotics, and autonomous agents.

Core Components:

- Memory
- Planning
- Tool-use (plugins, APIs)
- Feedback-driven behavior

What is Agentic AI?

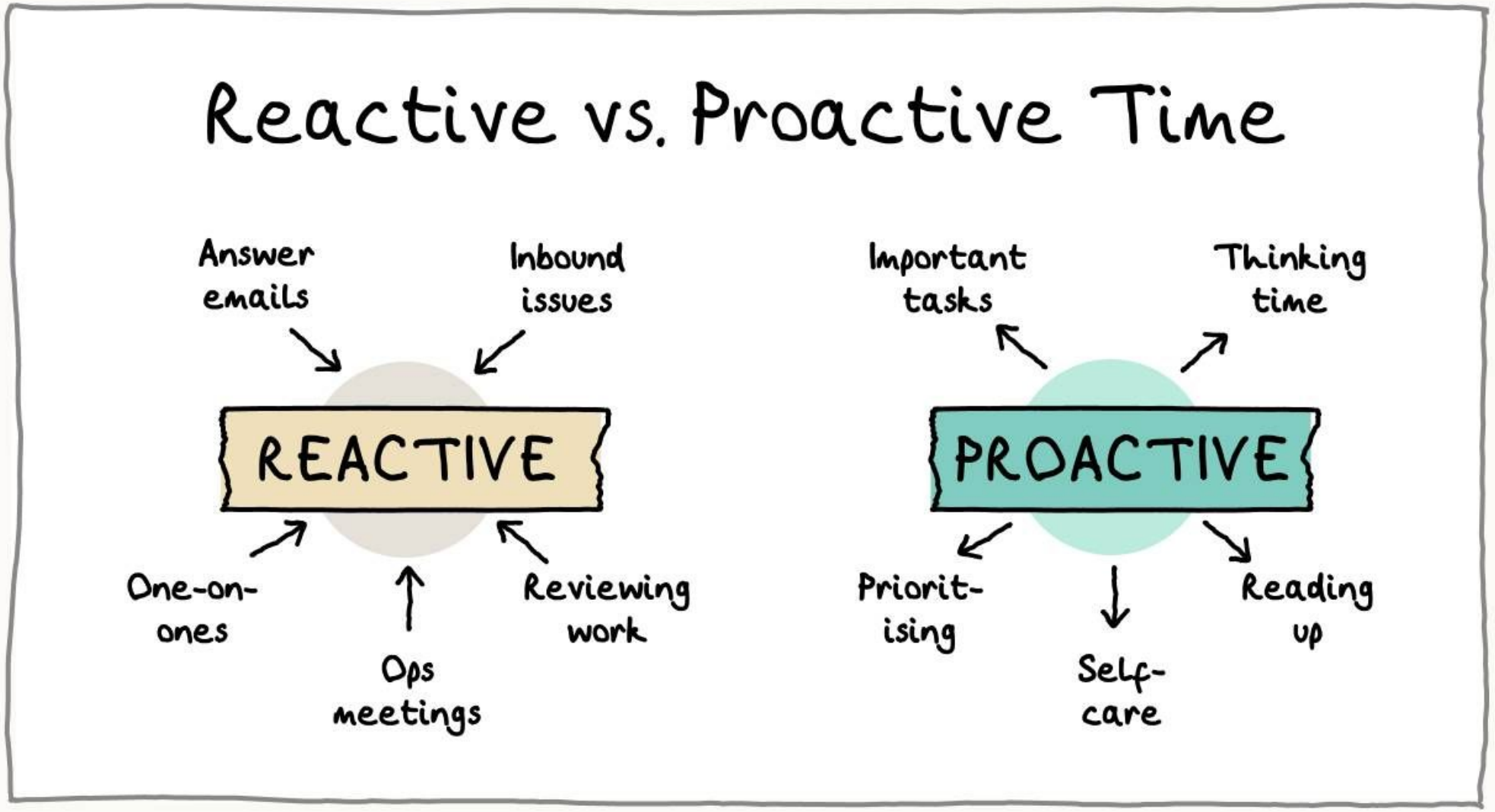


Reactive vs. Proactive Agents

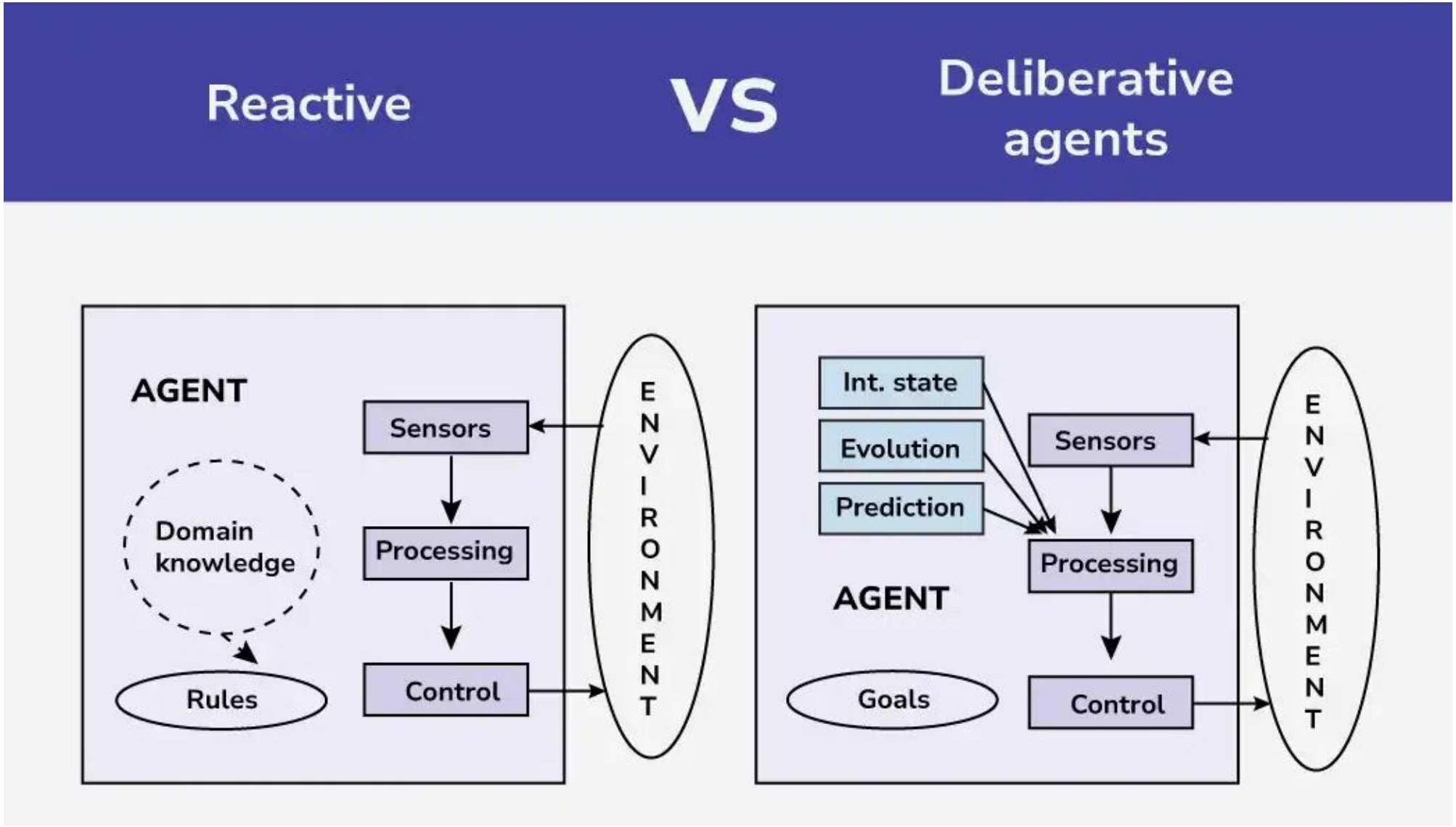
Feature	Reactive	Proactive
Response	Immediate	Goal-oriented
Planning	None	Yes
Learning	Event-triggered	Self-initiated
Example	Chatbots	Auto-GPT, Personal Assistants

Proactive agents generate goals, schedule actions, and evaluate results even without user prompts.

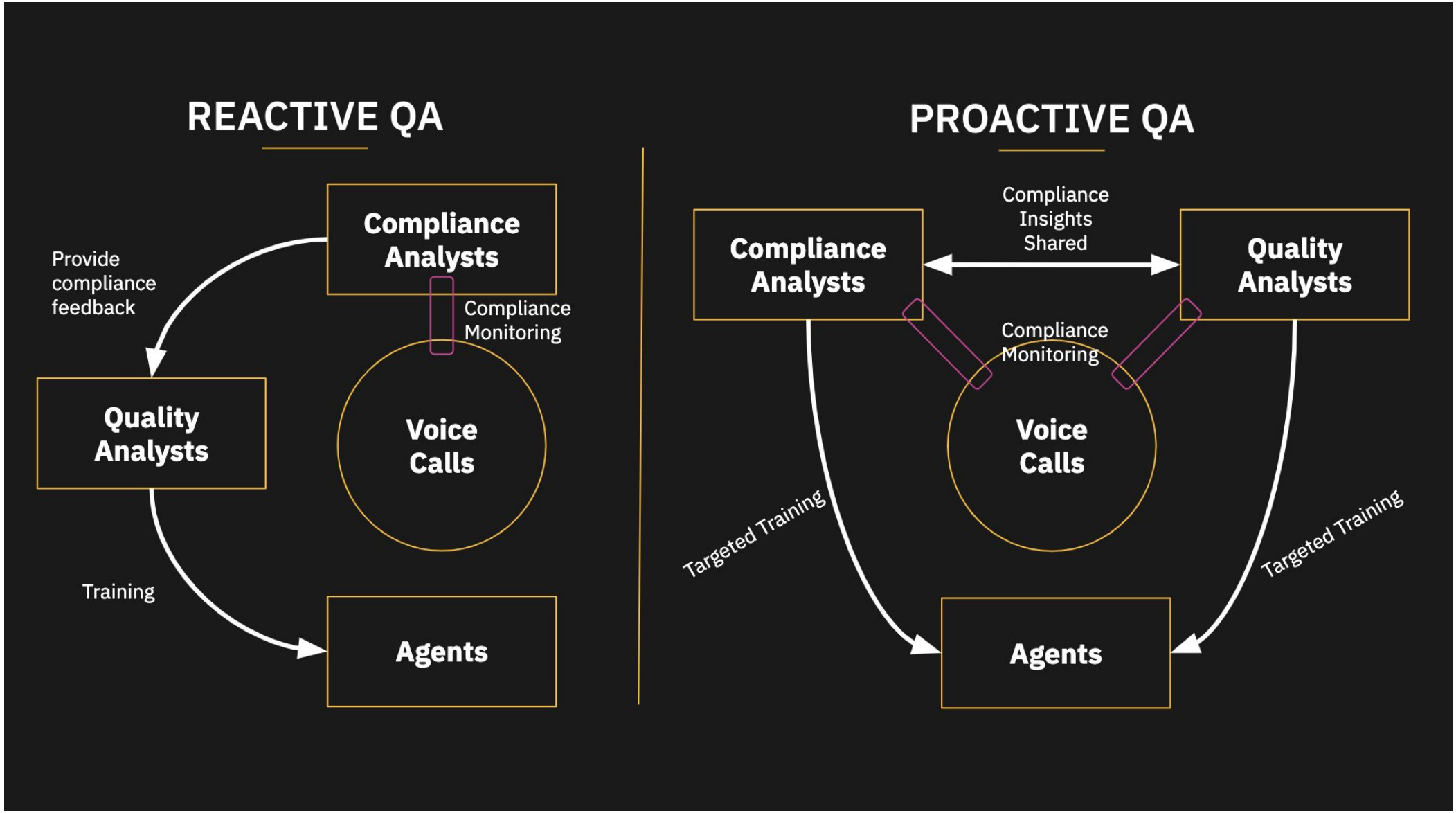
Reactive vs. Proactive Agents



Reactive vs. Proactive Agents



Reactive vs. Proactive Agents



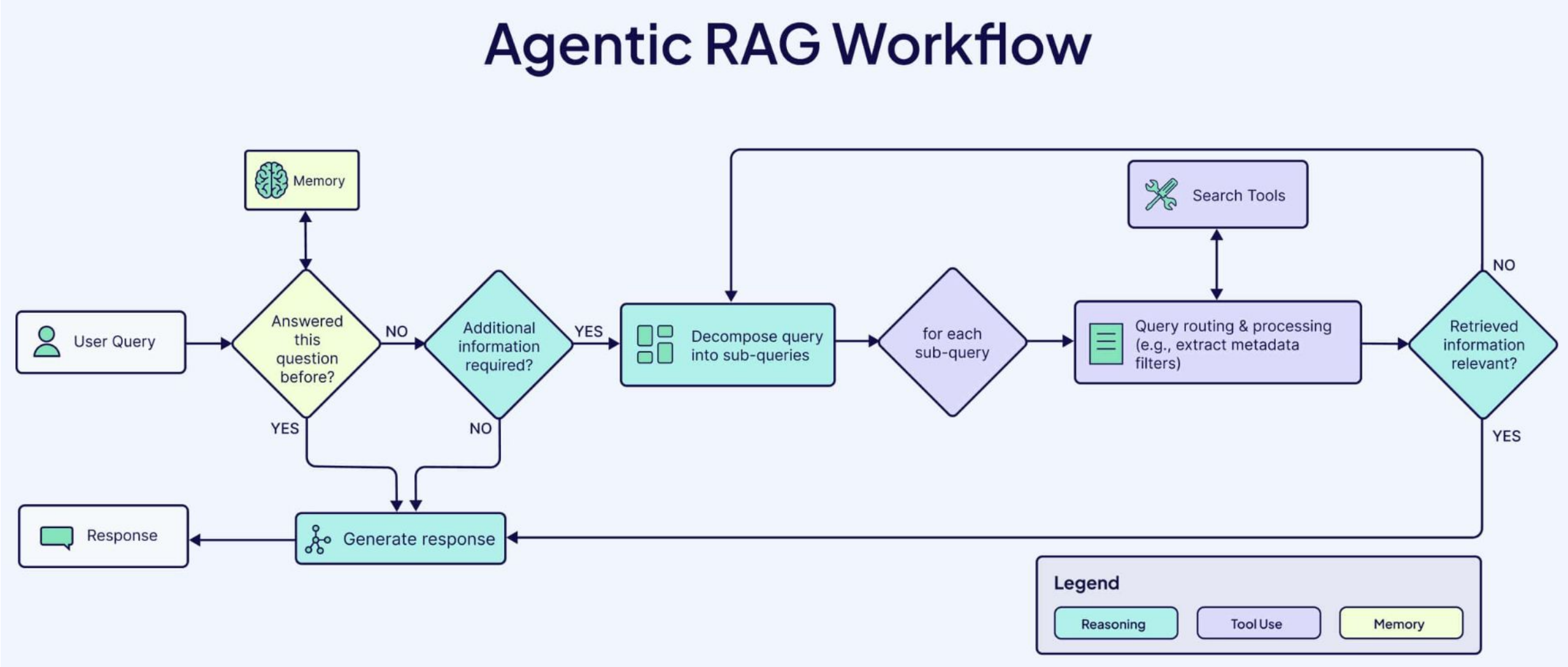
Agent Loop in Detail

- **Perceive:** Multimodal input (text, vision, memory)
- **Interpret:** NLP/vision models
- **Plan:** Task decomposition (e.g., LangChain chains)
- **Act:** API/tool calling
- **Reflect & Learn:** Refine memory or adjust goals

Applications:

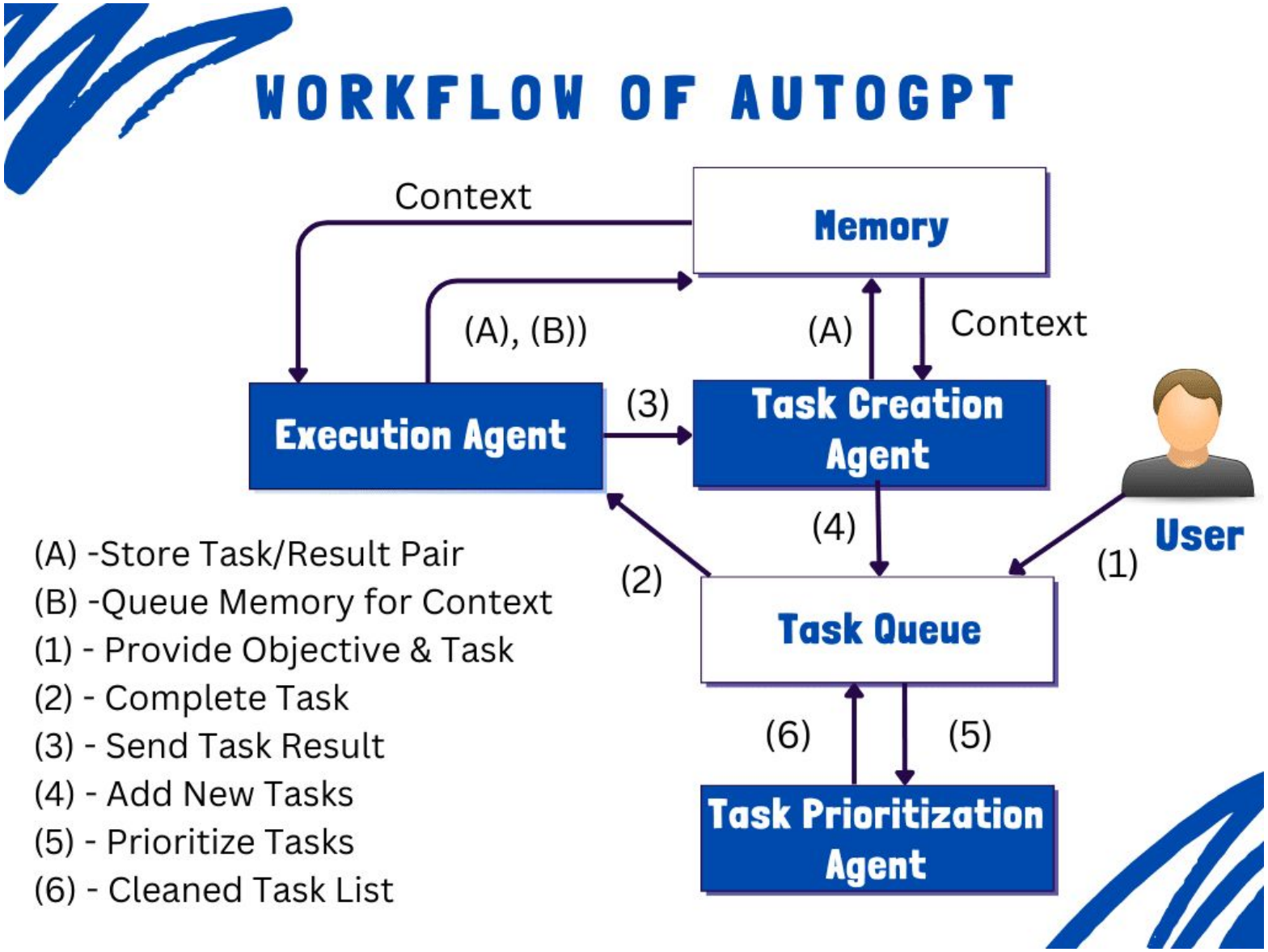
- Personal AI assistants
- Research co-pilots
- Automated workflows

Agent Loop in Detail



Open-Source Agentic Frameworks

Auto-GPT


WORKFLOW OF AUTOGPT
Context
Memory
(A), (B))
(A)
Context
Execution Agent
(3)
Task Creation Agent
User
(4)
(2)
(1)
Task Queue
(6)
(5)
Task Prioritization Agent
(A) -Store Task/Result Pair
(B) -Queue Memory for Context
(1) - Provide Objective & Task
(2) - Complete Task
(3) - Send Task Result
(4) - Add New Tasks
(5) - Prioritize Tasks
(6) - Cleaned Task List

Auto-GPT

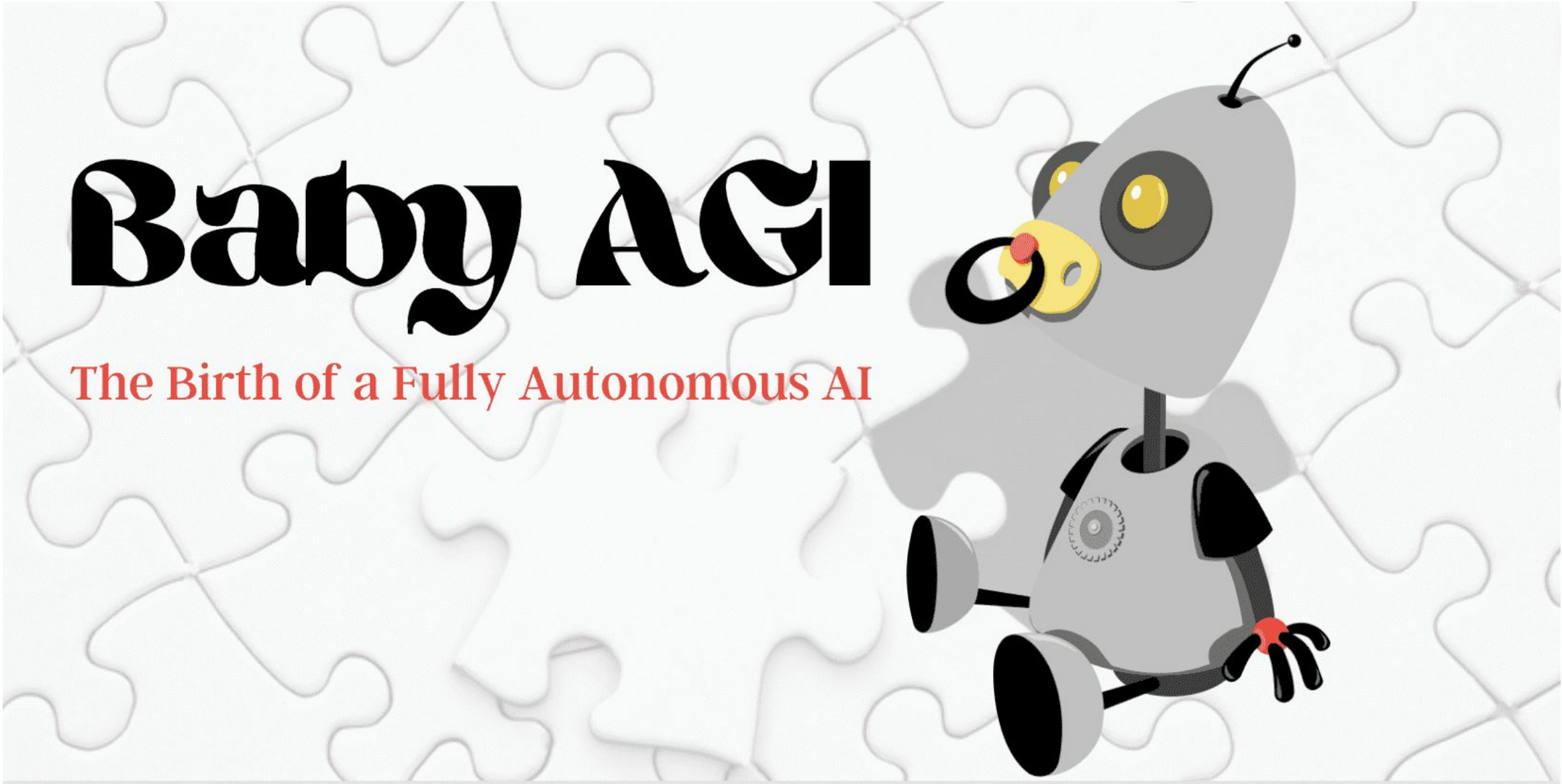
- **Open-source project** that chains GPT calls to build autonomous agents.
- Uses **long-term memory**, file storage, and self-generated tasks.
- **Requirements:**
  - Task input
  - Internet / API access
  - Feedback loop

GitHub: https://github.com/Torantulino/Auto-GPT

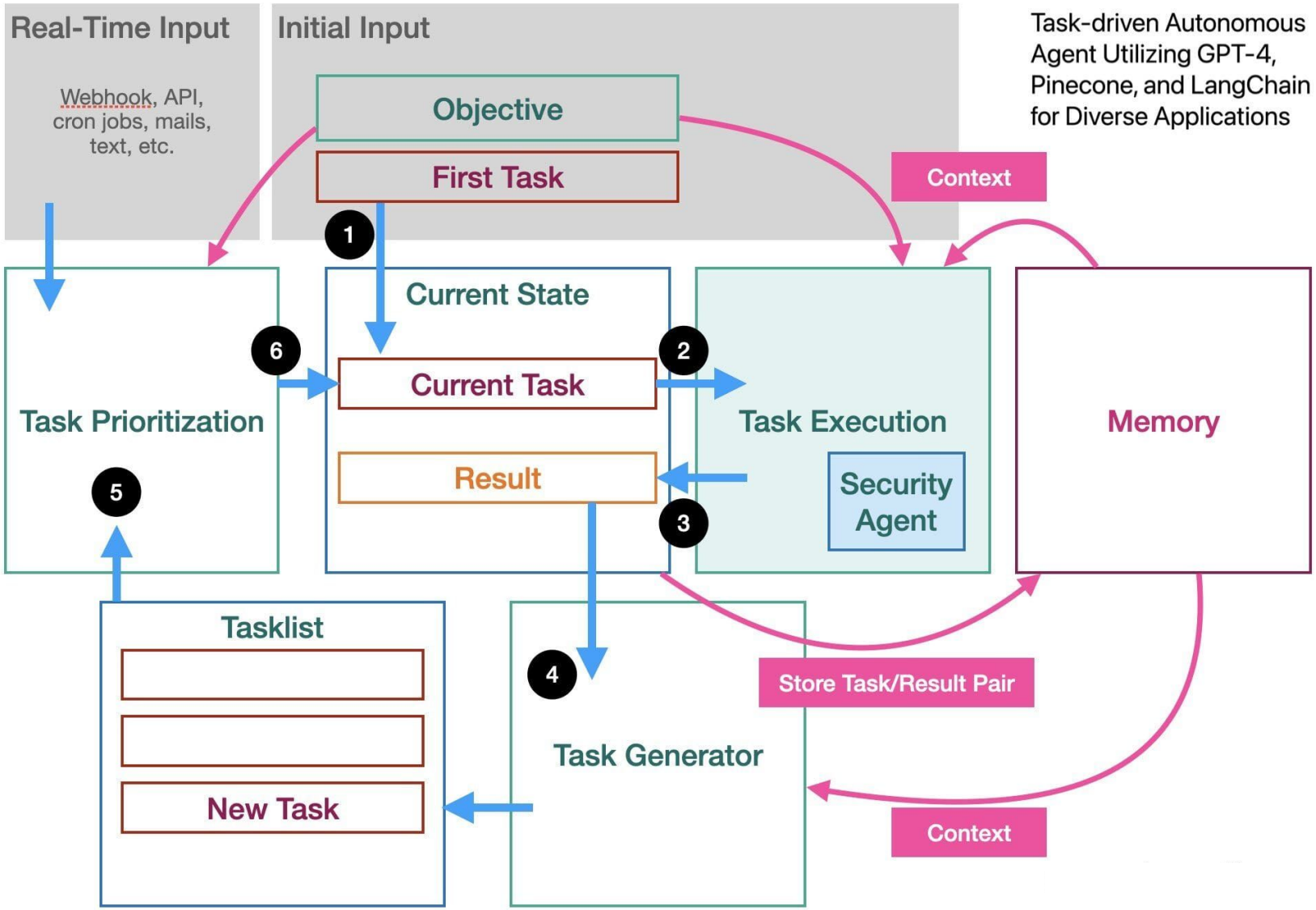
Auto-GPT


Auto GPT
VS
ChatGPT
An advanced version of GPT that is autonomous and self-learning
Description
A language model designed for conversational interfaces
Generates human-like text, learns from own mistakes
Features
Mimics human conversation, can engage on various topics
Can generate biased/ offensive content, lacks emotional intelligence
Limitations
Requires human intervention to improve performance
Generating high-quality text for businesses and organizations
Best suited for
Conversational interfaces and chatbots, customer service
blockchain-council.org

BabyAGI



BabyAGI

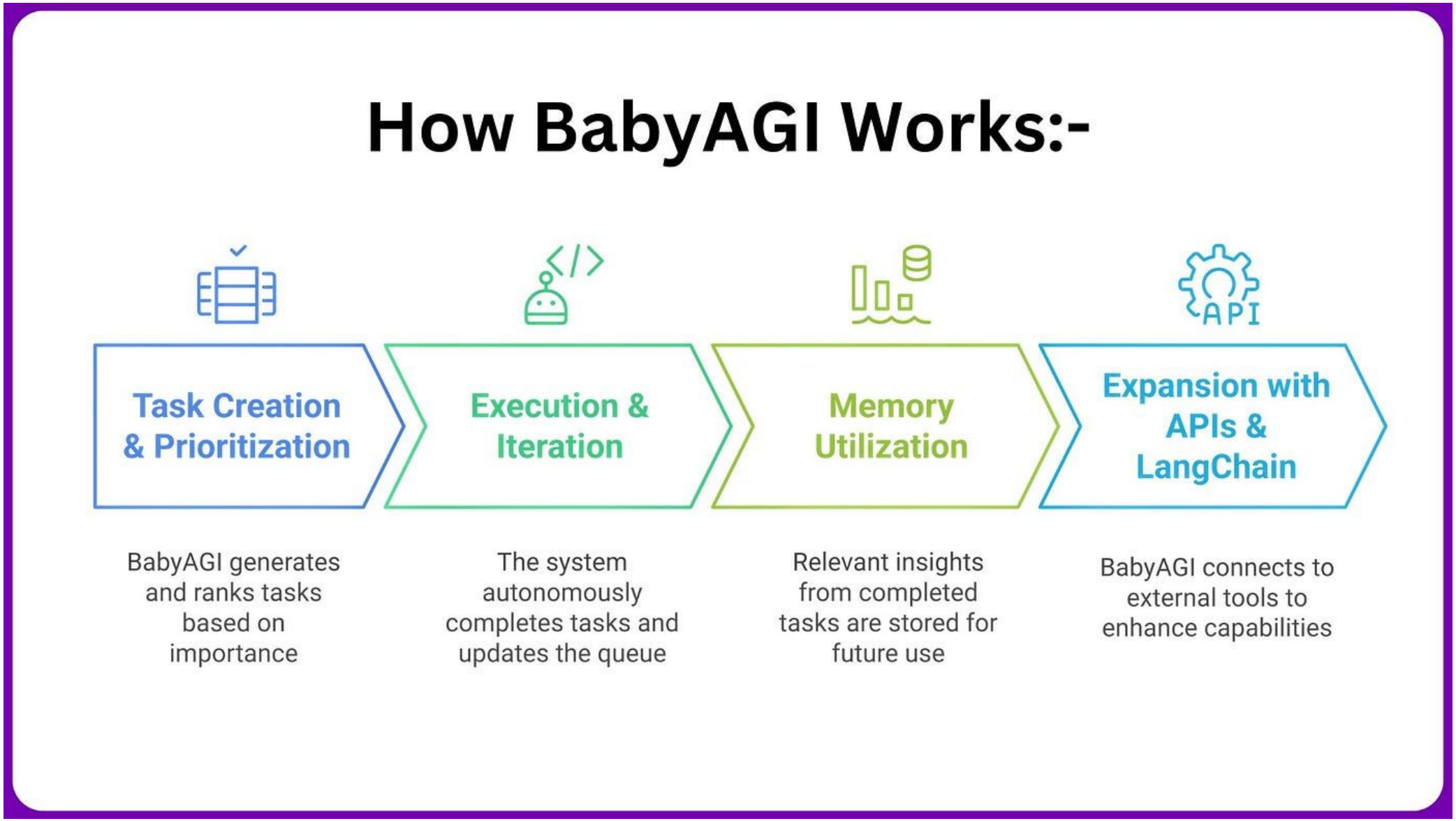

Real-Time Input
Webhook, API, cron jobs, mails, text, etc.
Initial Input
Objective
First Task
Task-driven Autonomous Agent Utilizing GPT-4, Pinecone, and LangChain for Diverse Applications
Context
1
Current State
6
2
Current Task
Task Prioritization
Task Execution
Memory
Result
Security Agent
5
3
Tasklist
4
Store Task/Result Pair
Task Generator
New Task
Context

BabyAGI

- **Lightweight Python agent loop**
- Generates tasks from high-level objectives
- Runs tasks, stores results, and generates new tasks iteratively
- Emphasizes simplicity and minimalism

GitHub: https://github.com/yoheinakajima/babyagi

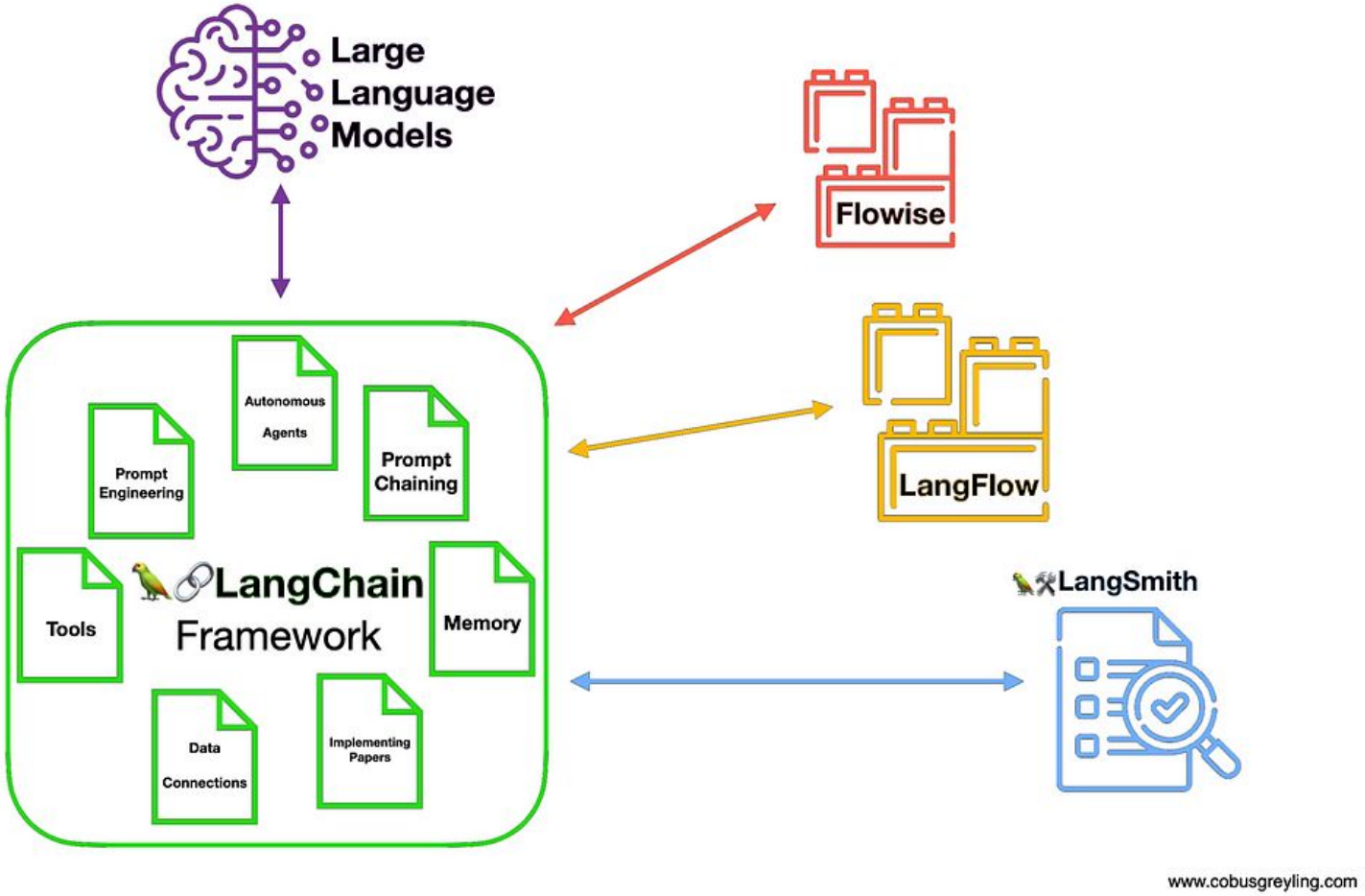
BabyAGI


How BabyAGI Works:-
Task Creation & Prioritization
Execution & Iteration
Memory Utilization
Expansion with APIs & LangChain
API
BabyAGI generates and ranks tasks based on importance
The system autonomously completes tasks and updates the queue
Relevant insights from completed tasks are stored for future use
BabyAGI connects to external tools to enhance capabilities

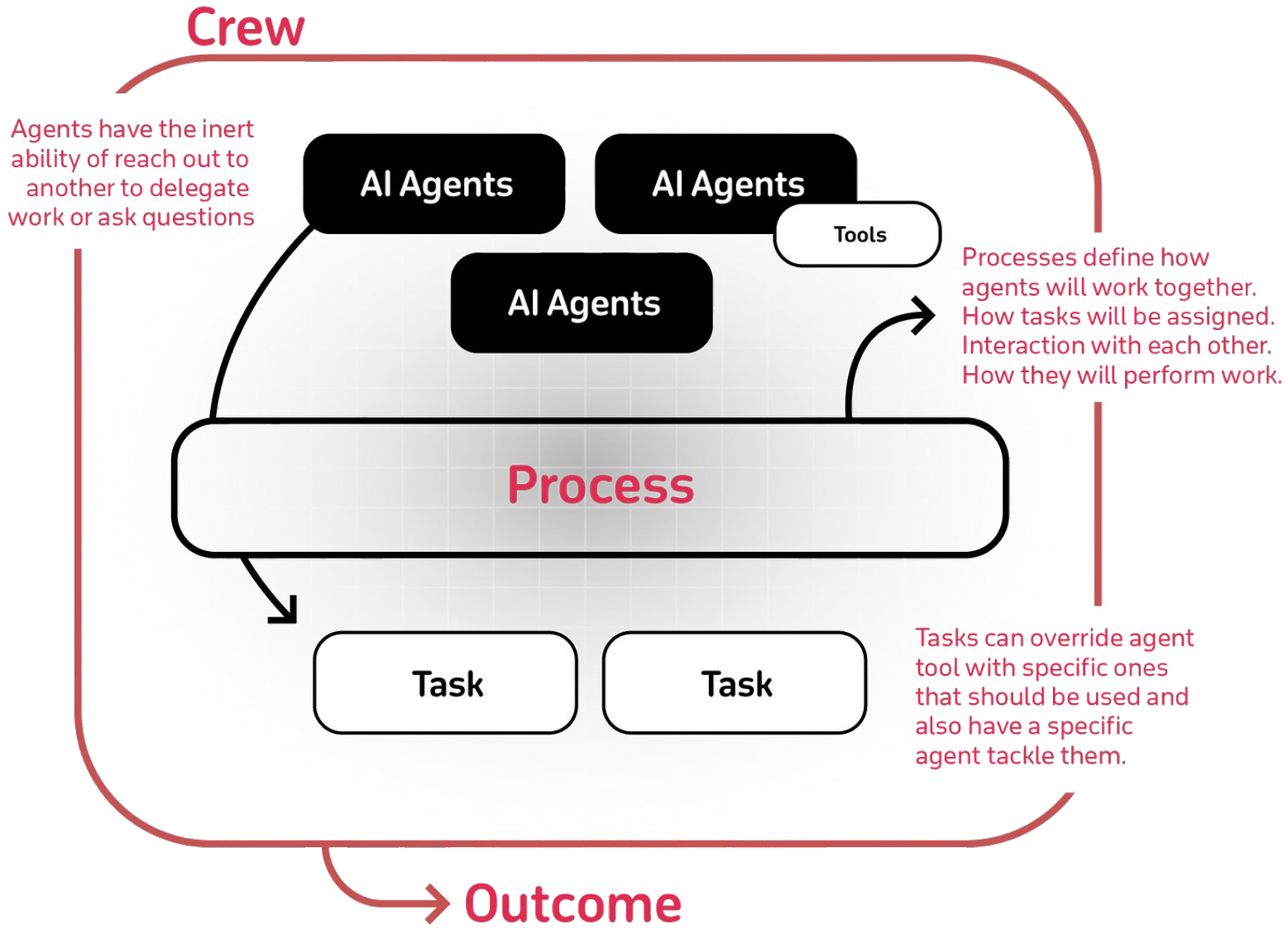
Other Notable Frameworks

- **LangChain:** Framework for chaining LLMs with tools, APIs, and memory. Enables complex workflows and agentic behavior.
- **CrewAI:** Multi-agent coordination platform for collaborative task execution.
- **SuperAGI:** Production-grade agent platform supporting extensibility, monitoring, and deployment.
- **OpenDevin:** Developer-oriented agentic IDE for automating software engineering tasks.

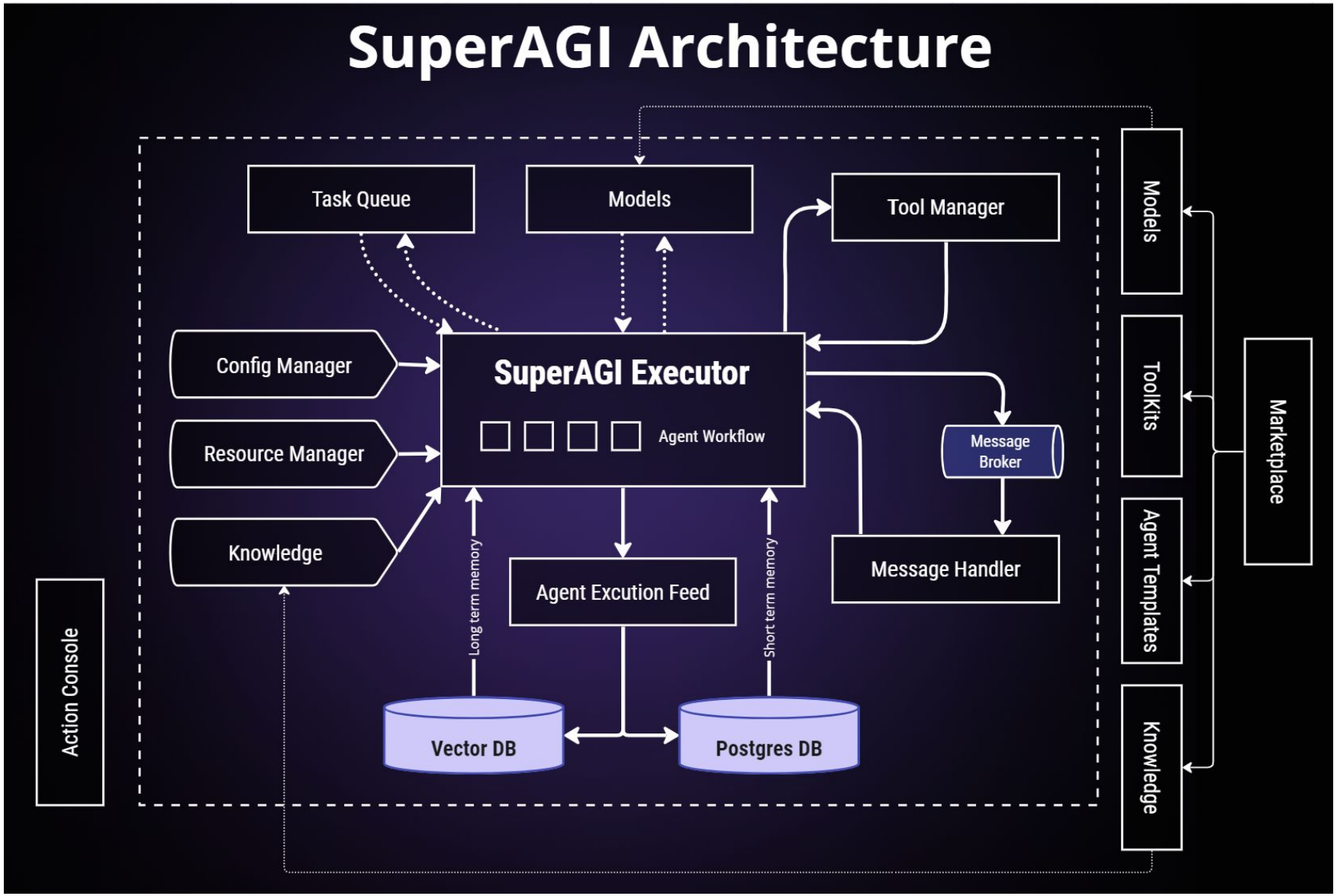
LangChain Ecosystem

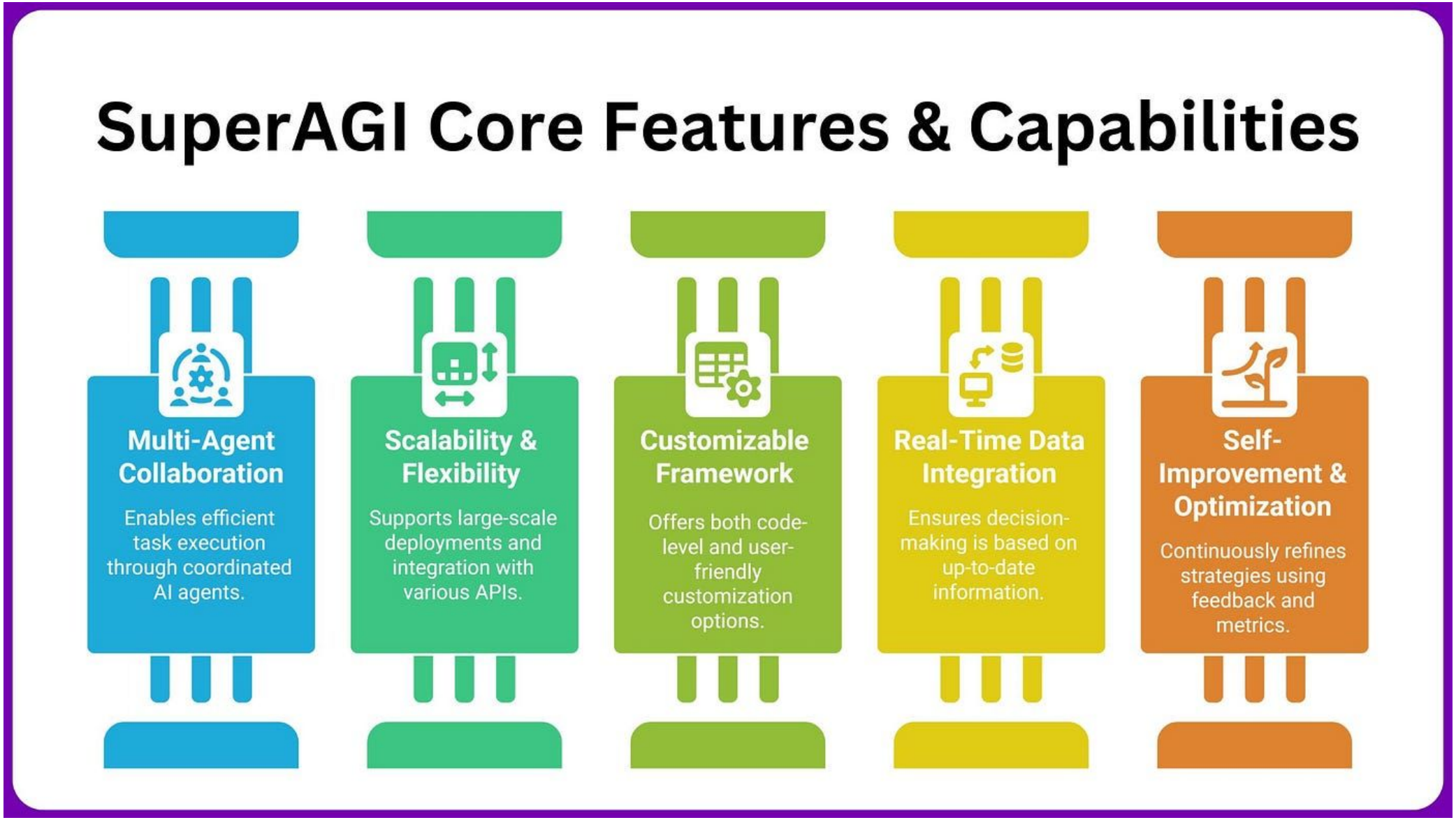


CrewAI

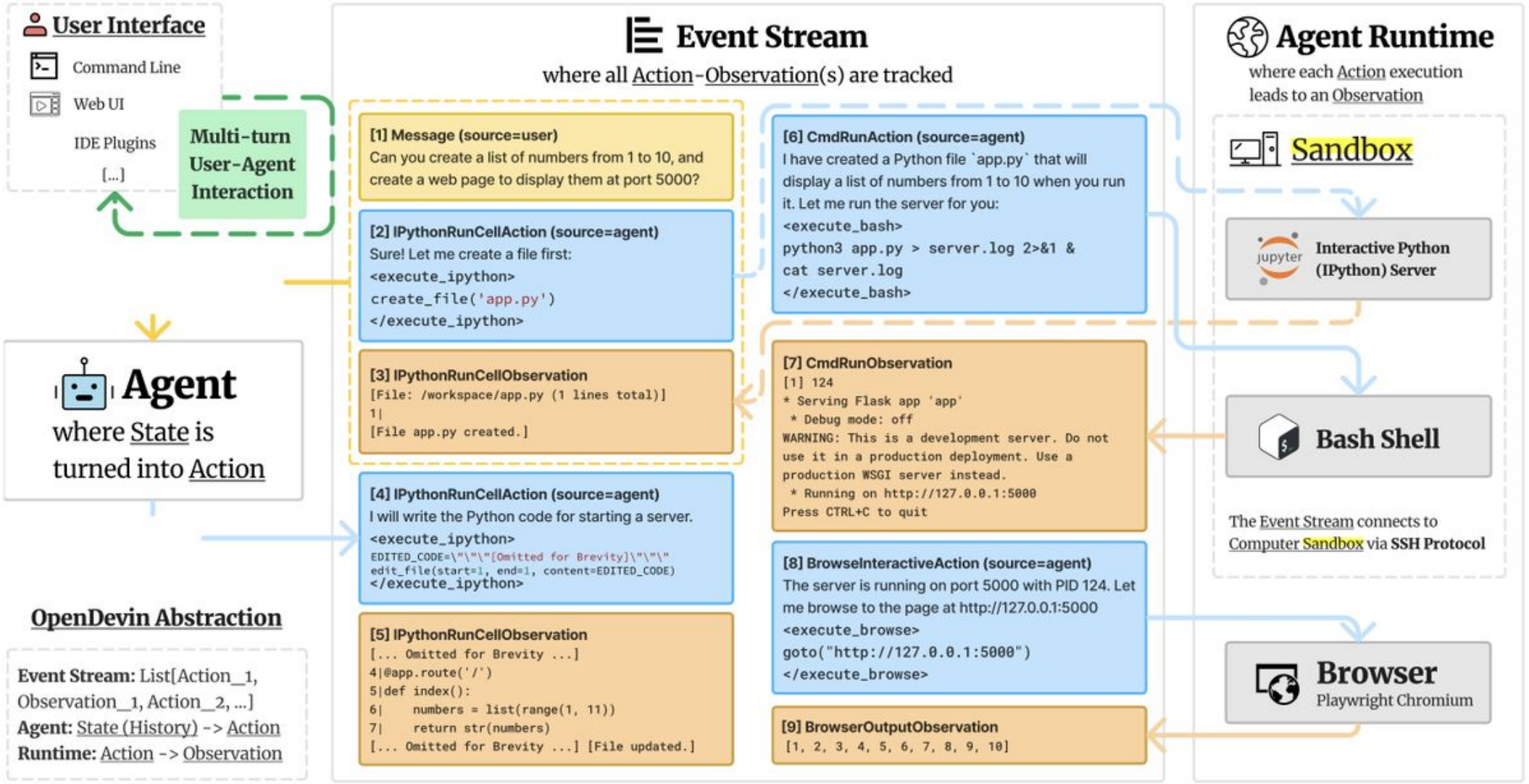

Crew
Agents have the inert ability of reach out to another to delegate work or ask questions
AI Agents
AI Agents
Tools
AI Agents
Processes define how agents will work together.
How tasks will be assigned.
Interaction with each other.
How they will perform work.
Process
Task
Task
Tasks can override agent tool with specific ones that should be used and also have a specific agent tackle them.
Outcome

SuperAGI

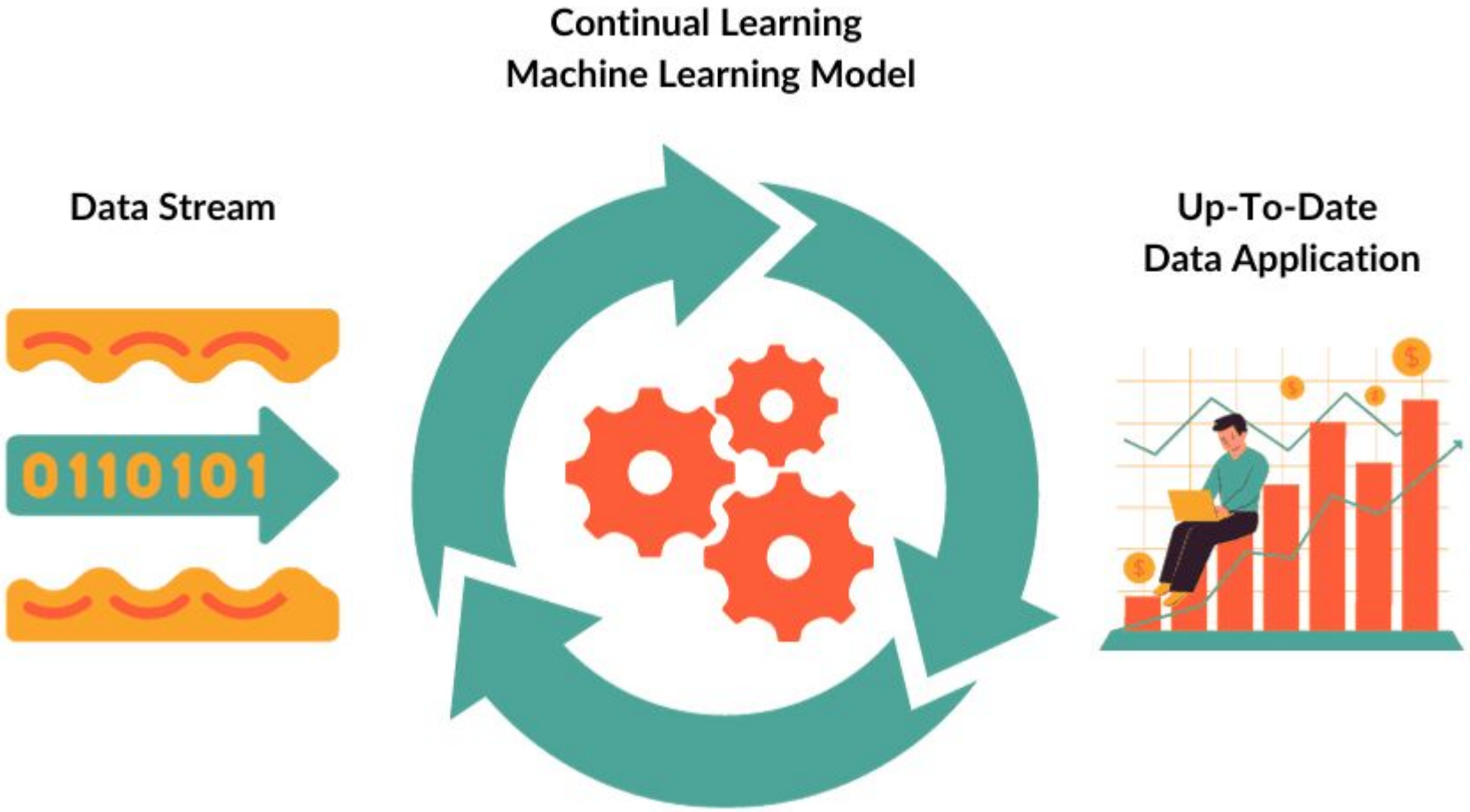

SuperAGI Architecture
Task Queue
Models
Tool Manager
Config Manager
SuperAGI Executor
Agent Workflow
Resource Manager
Message Broker
Knowledge
Long term memory
Agent Excution Feed
Short term memory
Message Handler
Vector DB
Postgres DB
Action Console
Models
ToolKits
Agent Templates
Knowledge
Marketplace


SuperAGI Core Features & Capabilities
Multi-Agent Collaboration
Enables efficient task execution through coordinated AI agents.
Scalability & Flexibility
Supports large-scale deployments and integration with various APIs.
Customizable Framework
Offers both code-level and user-friendly customization options.
Real-Time Data Integration
Ensures decision-making is based on up-to-date information.
Self-Improvement & Optimization
Continuously refines strategies using feedback and metrics.

OpenDevin


User Interface
Command Line
Web UI
IDE Plugins
[...]
Multi-turn User-Agent Interaction
Agent
where State is turned into Action
OpenDevin Abstraction
Event Stream: List[Action_1, Observation_1, Action_2, ...]
Agent: State (History) -> Action
Runtime: Action -> Observation
Event Stream
where all Action-Observation(s) are tracked
[1] Message (source=user)
Can you create a list of numbers from 1 to 10, and create a web page to display them at port 5000?
[2] IPythonRunCellAction (source=agent)
Sure! Let me create a file first:
<execute_ipython>
create_file('app.py')
</execute_ipython>
[3] IPythonRunCellObservation
[File: /workspace/app.py (1 lines total)]
1|
[File app.py created.]
[4] IPythonRunCellAction (source=agent)
I will write the Python code for starting a server.
<execute_ipython>
EDITED_CODE=\"\"\"[Omitted for Brevity]\"\"\"
edit_file(start=1, end=1, content=EDITED_CODE)
</execute_ipython>
[5] IPythonRunCellObservation
[... Omitted for Brevity ...]
4|@app.route('/')
5|def index():
6| numbers = list(range(1, 11))
7| return str(numbers)
[... Omitted for Brevity ...] [File updated.]
[6] CmdRunAction (source=agent)
I have created a Python file `app.py` that will display a list of numbers from 1 to 10 when you run it. Let me run the server for you:
<execute_bash>
python3 app.py > server.log 2>&1 &
cat server.log
</execute_bash>
[7] CmdRunObservation
[1] 124
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
[8] BrowseInteractiveAction (source=agent)
The server is running on port 5000 with PID 124. Let me browse to the page at http://127.0.0.1:5000
<execute_browse>
goto("http://127.0.0.1:5000")
</execute_browse>
[9] BrowserOutputObservation
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Agent Runtime
where each Action execution leads to an Observation
Sandbox
Interactive Python (IPython) Server
Bash Shell
The Event Stream connects to Computer Sandbox via SSH Protocol
Browser
Playwright Chromium

Continual Learning in Agents


Continual Learning
Machine Learning Model
Data Stream
0110101
Up-To-Date
Data Application

Why Continual Learning?

- Agentic AI should evolve with time:
  - Learn from feedback
  - Adapt to new tasks
  - Retain useful knowledge

Real-world scenarios:

- Personalization
- Environment drift
- Avoiding forgetting

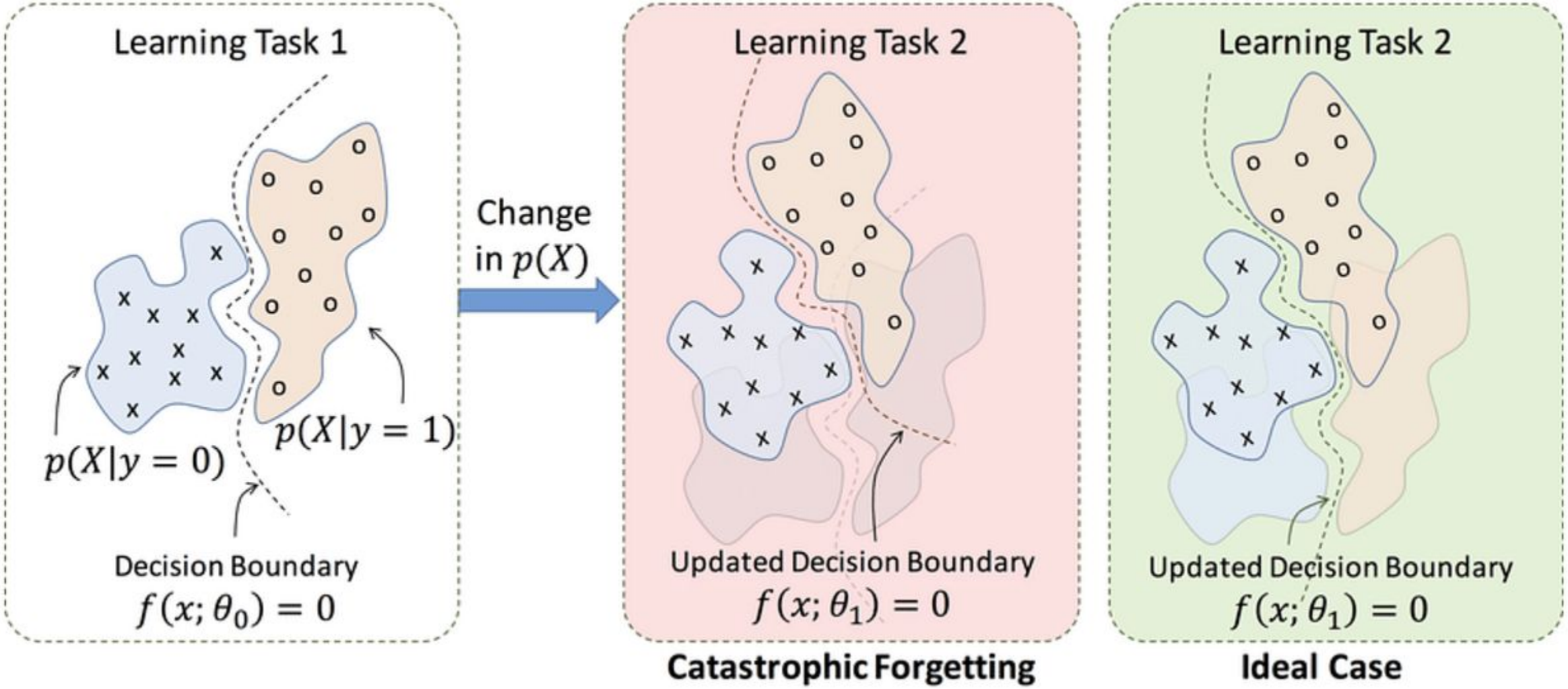
Challenges in Continual Learning

- **Catastrophic forgetting:** Agents may lose previously acquired knowledge when learning new tasks.
- **Stability-plasticity dilemma:** Balancing the ability to learn new information (plasticity) with retaining old knowledge (stability).
- **Modality drift:** Changes or shifts in the types or distributions of input modalities over time.
- **Scaling memory:** Managing both semantic (general knowledge) and episodic (specific experiences) memory as agents interact with the world.

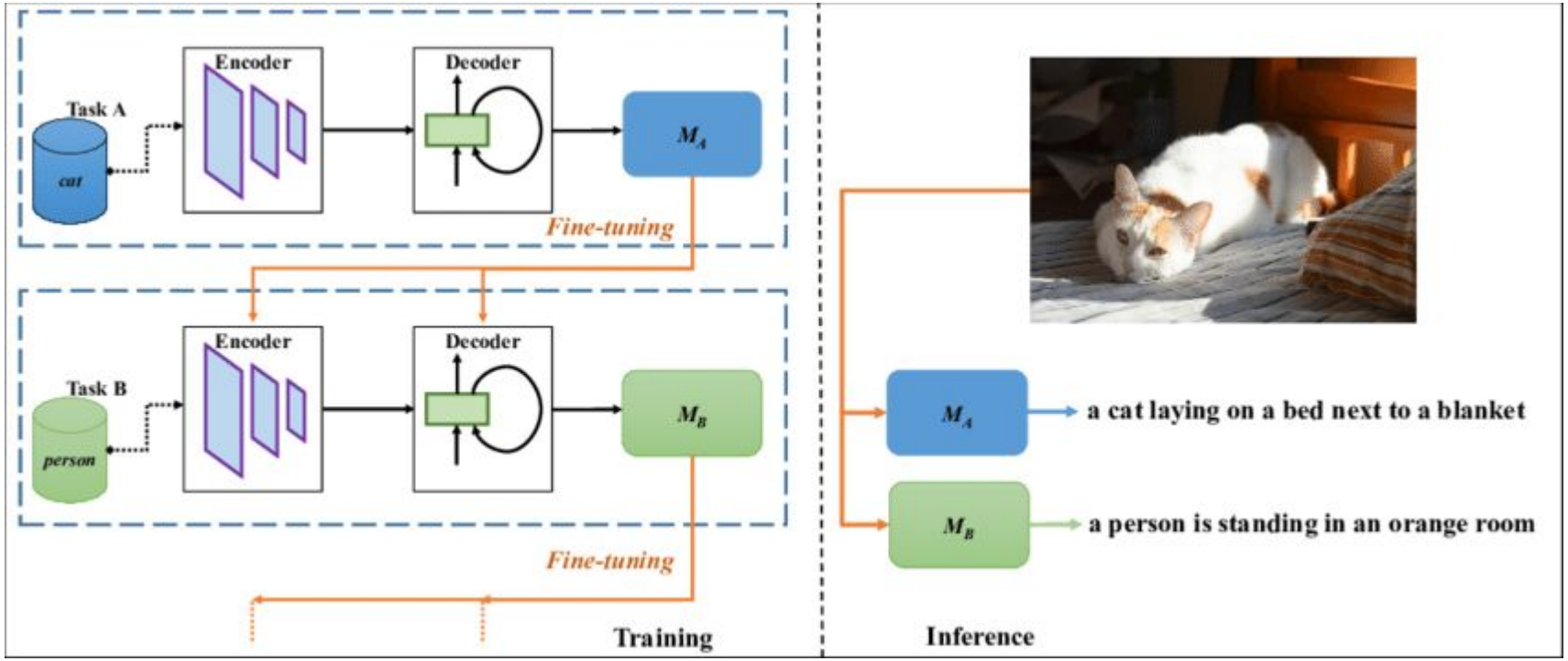
Solutions:

- Replay memory (experience replay)
- Progressive networks
- Elastic weight consolidation (EWC)

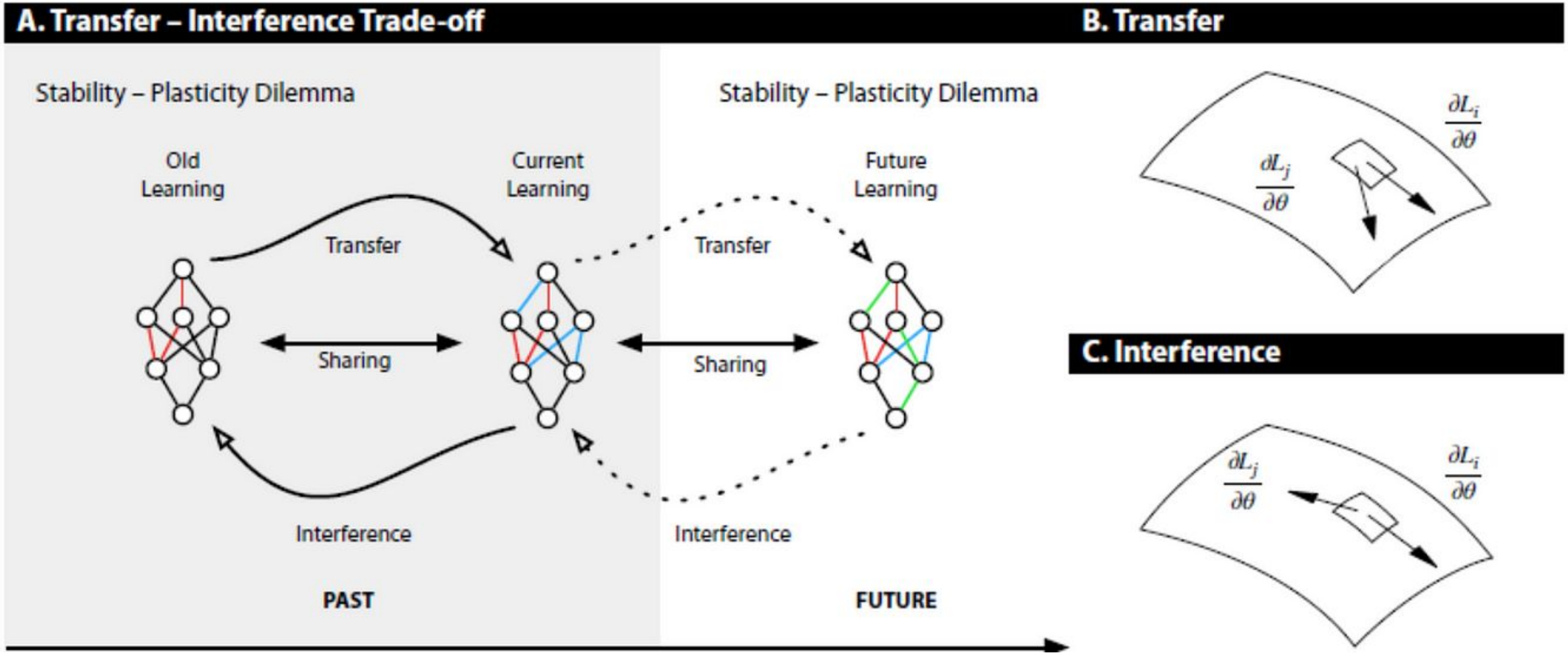
Catastrophic forgetting



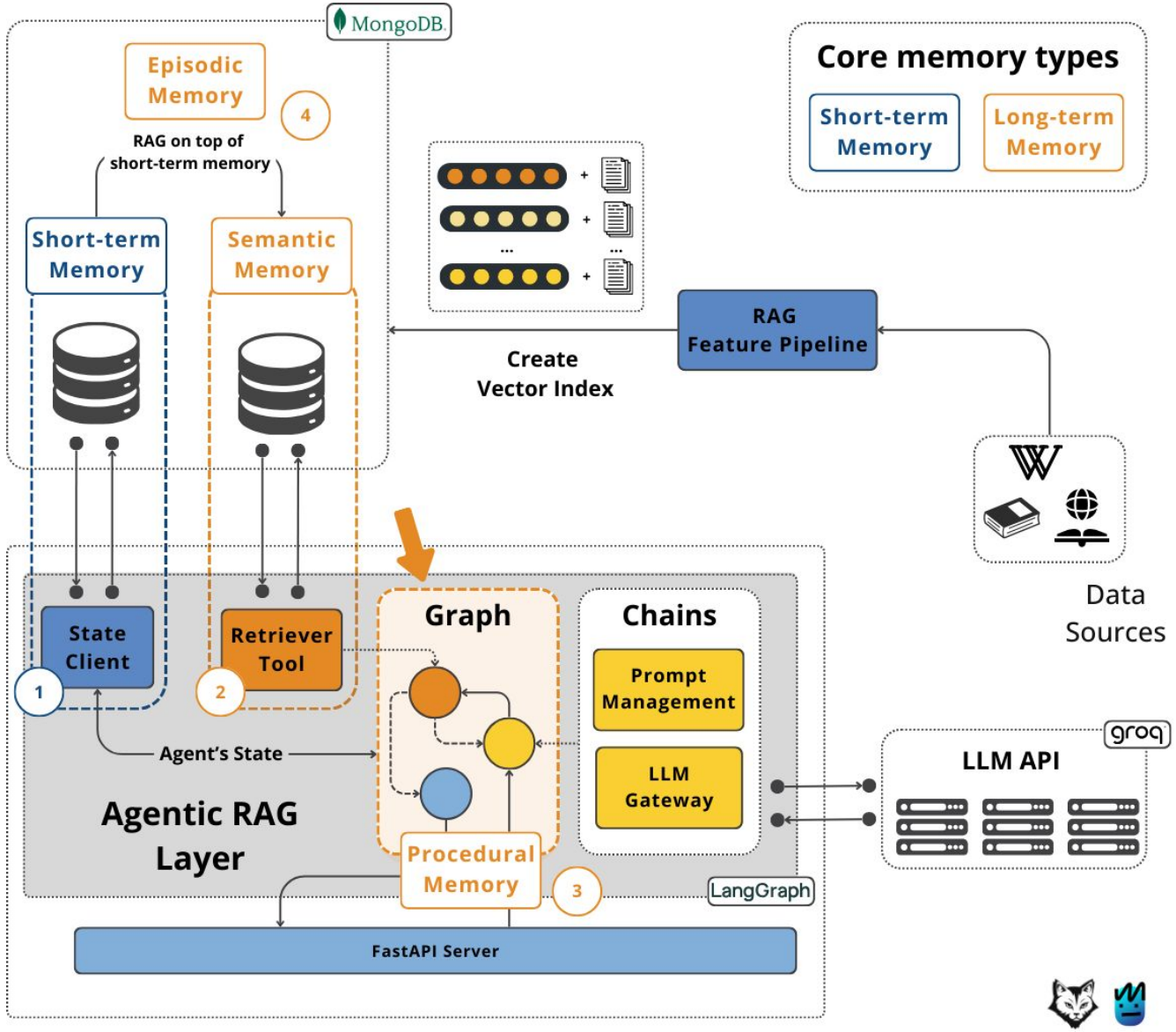
Catastrophic forgetting



Stability-plasticity dilemma



Scaling memory



Safety, Ethics Control

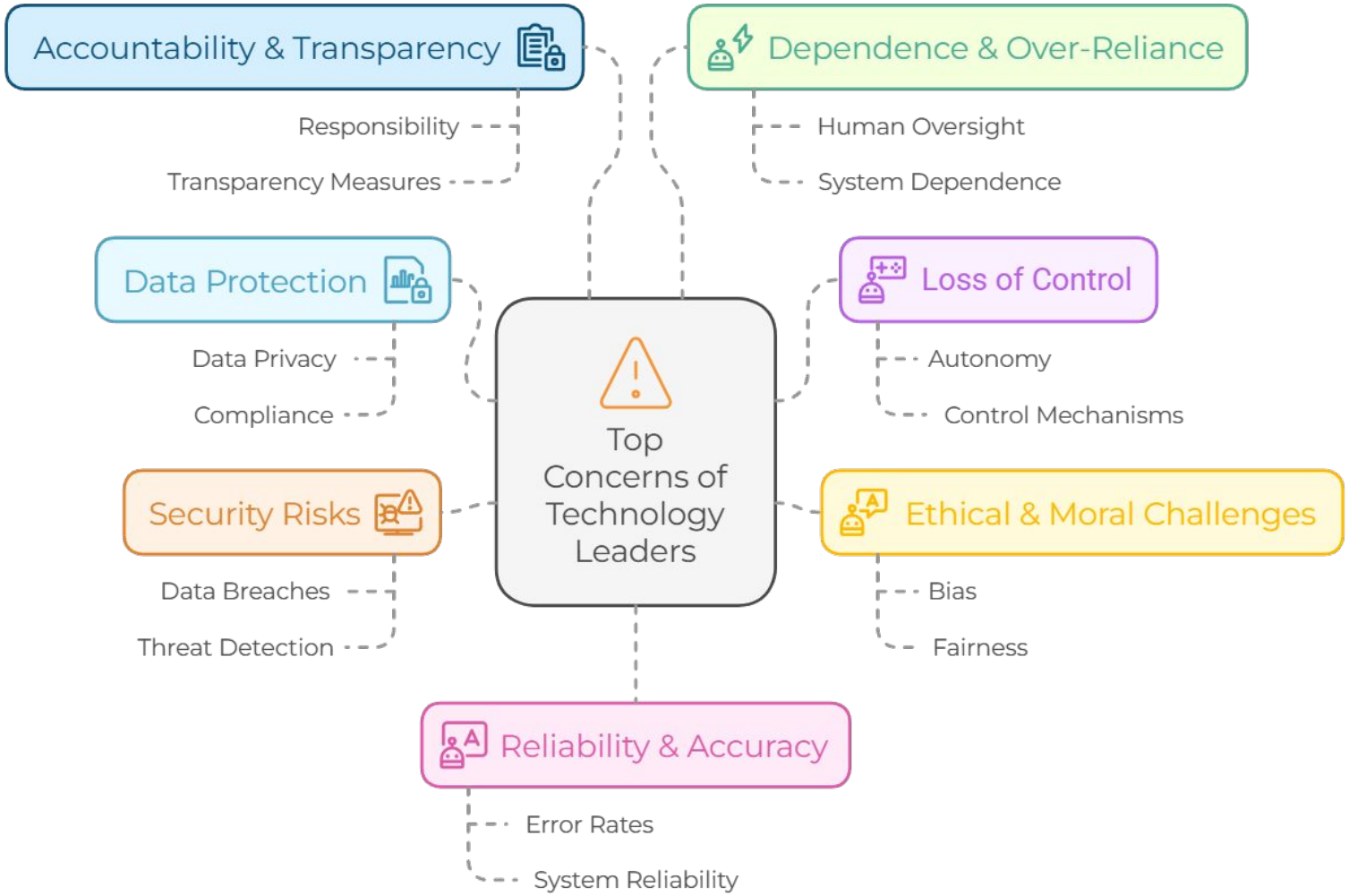
Safety in Agentic Systems

- **Risks:**
  - Autonomous actions without human oversight
  - Errors in tool or API use
  - Prompt injection or goal hijacking
  - Recursive self-improvement without constraints

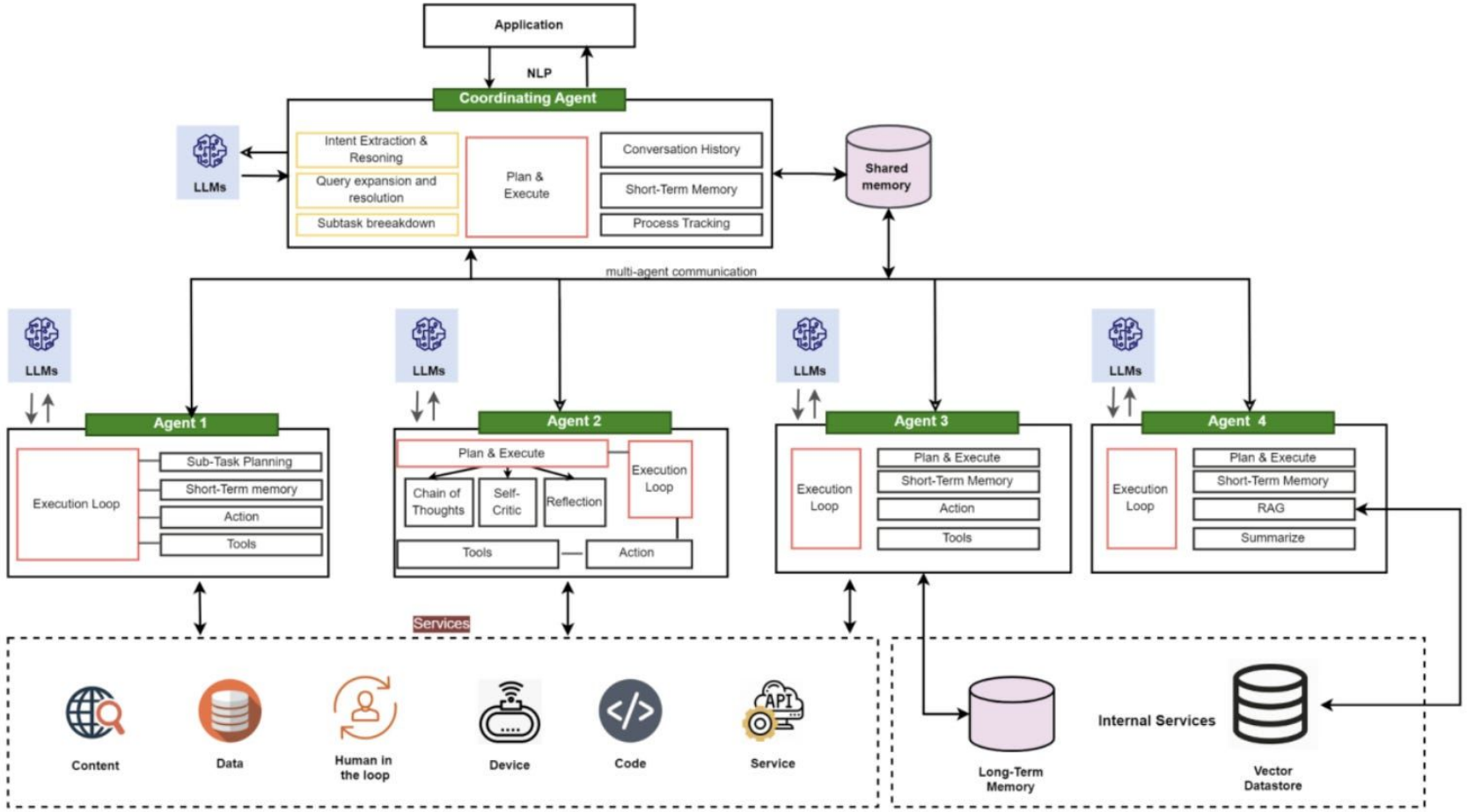
Mitigations:

- Approval-based execution (human-in-the-loop)
- Guardrails and policy enforcement
- Explainability (XAI) for agent decisions

Safety in Agentic Systems

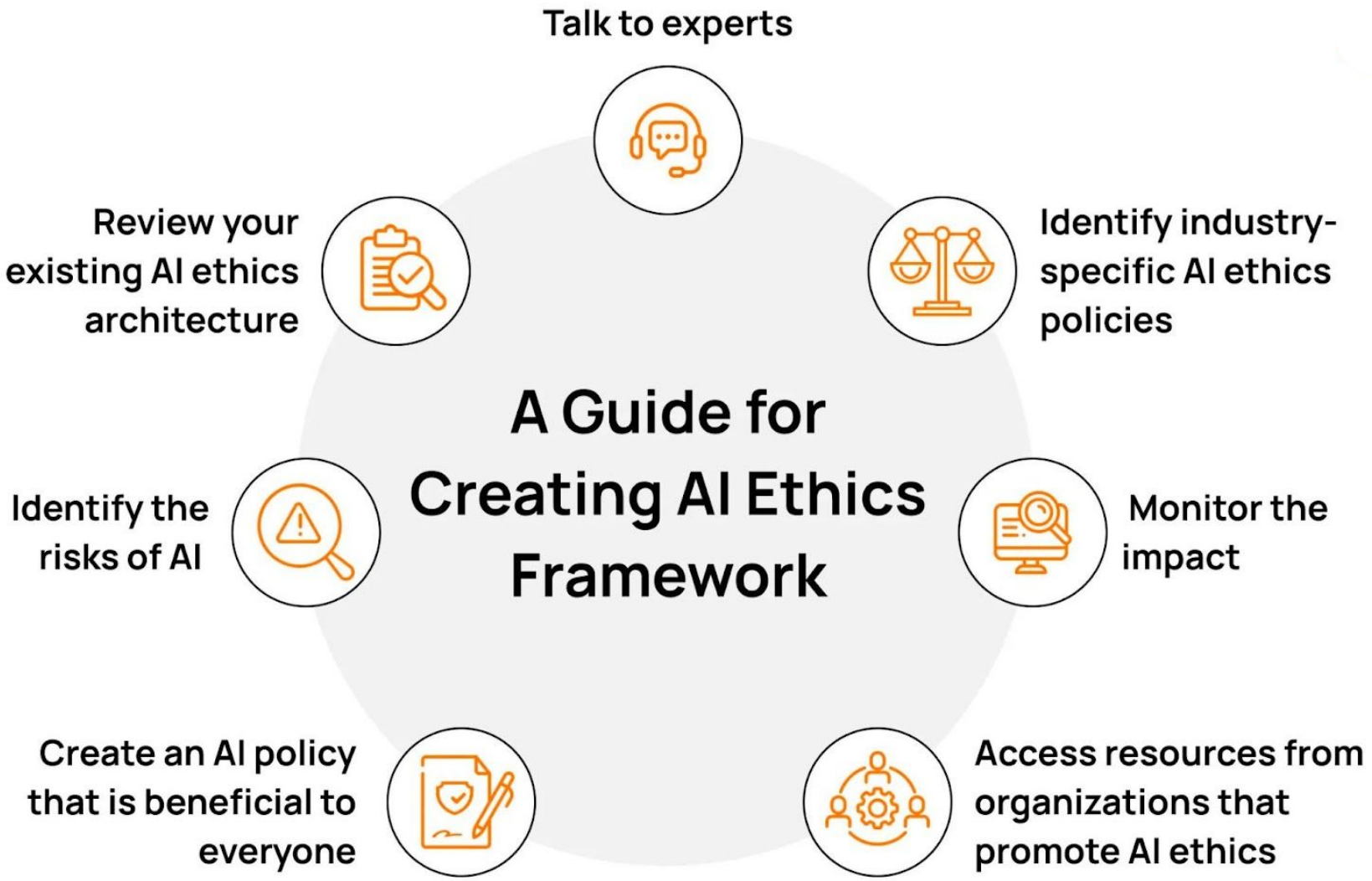


Safety in Agentic Systems



The diagram depicts an *example of* multi-agent architecture of specialized agent functionality. Specialized functionality is a form of agentic patterns and could be exhibited by any agent depending on the use case.

Ethics and Control



Ethics and Control

- **Alignment:** Are agents acting in users' interests?
- **Accountability:** Who is responsible for AI actions?
- **Control:** Can humans override or audit decisions?

Key Areas:

- Fairness in multimodal data
- Bias amplification (vision + language)
- Transparency in decision-making

Future Directions

Open Problems and Research Directions

- **Multimodal grounding in real-world environments:** How can agents robustly connect language, vision, and action in dynamic, noisy settings?
- **World model integration:** Combining temporal and spatial memory for persistent, context-aware reasoning.
- **Agent collaboration:** Enabling effective communication and coordination in multi-agent systems.
- **Scaling memory-efficient agents:** Designing architectures that scale to long-term, large-scale memory without prohibitive costs.
- **Robust reward models:** Developing reliable reward and feedback mechanisms for open-ended, real-world tasks.
- **Meta-cognition:** Building agents that can introspect, self-monitor, and adapt their own reasoning processes.

Summary

- Multimodal models (CLIP, VisualBERT, FLAVA) integrate perception and language for richer AI understanding.
- Agentic AI enables proactive, goal-driven, and autonomous behavior.
- Open-source frameworks (Auto-GPT, BabyAGI) are accelerating research and applications.
- Ethical, safe, and explainable deployment is essential for real-world impact.
- The future of AI lies in continual learning and intelligent autonomy.

References

[1] Radford, A., Kim, J. W., Hallacy, C., et al. (2021).

CLIP: Learning Transferable Visual Models From Natural Language Supervision.

arXiv:2103.00020.

[2] Li, L. H., Yatskar, M., Yin, D., Hsieh, C. J., Chang, K. W. (2019).

VisualBERT: A Simple and Performant Baseline for Vision and Language.

arXiv:1908.03557.

[3] Singh, A., Goyal, N., Goswami, V., et al. (2021).

FLAVA: A Foundational Language And Vision Alignment Model.

arXiv:2112.04482.

[4] Yao, S., Zhao, J., Yu, D., et al. (2023).

ReAct: Synergizing Reasoning and Acting in Language Models.

arXiv:2210.03629.

References

[5] Nakajima, Y. (2023).

BabyAGI.

https://github.com/yoheinakajima/babyagi

[6] Torantulino (2023).

Auto-GPT.

https://github.com/Torantulino/Auto-GPT

[7] Ahn, M., Brohan, A., Chebotar, Y., et al. (2022).

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances.

arXiv:2204.01691.

[8] OpenAI Blog.

Agent Simulations and Superalignment.

https://openai.com/blog/

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.