RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Gormley Lab
www.gormleylab.com

AI 101: Machine Learning, Self-Driving Labs, and AI Agents

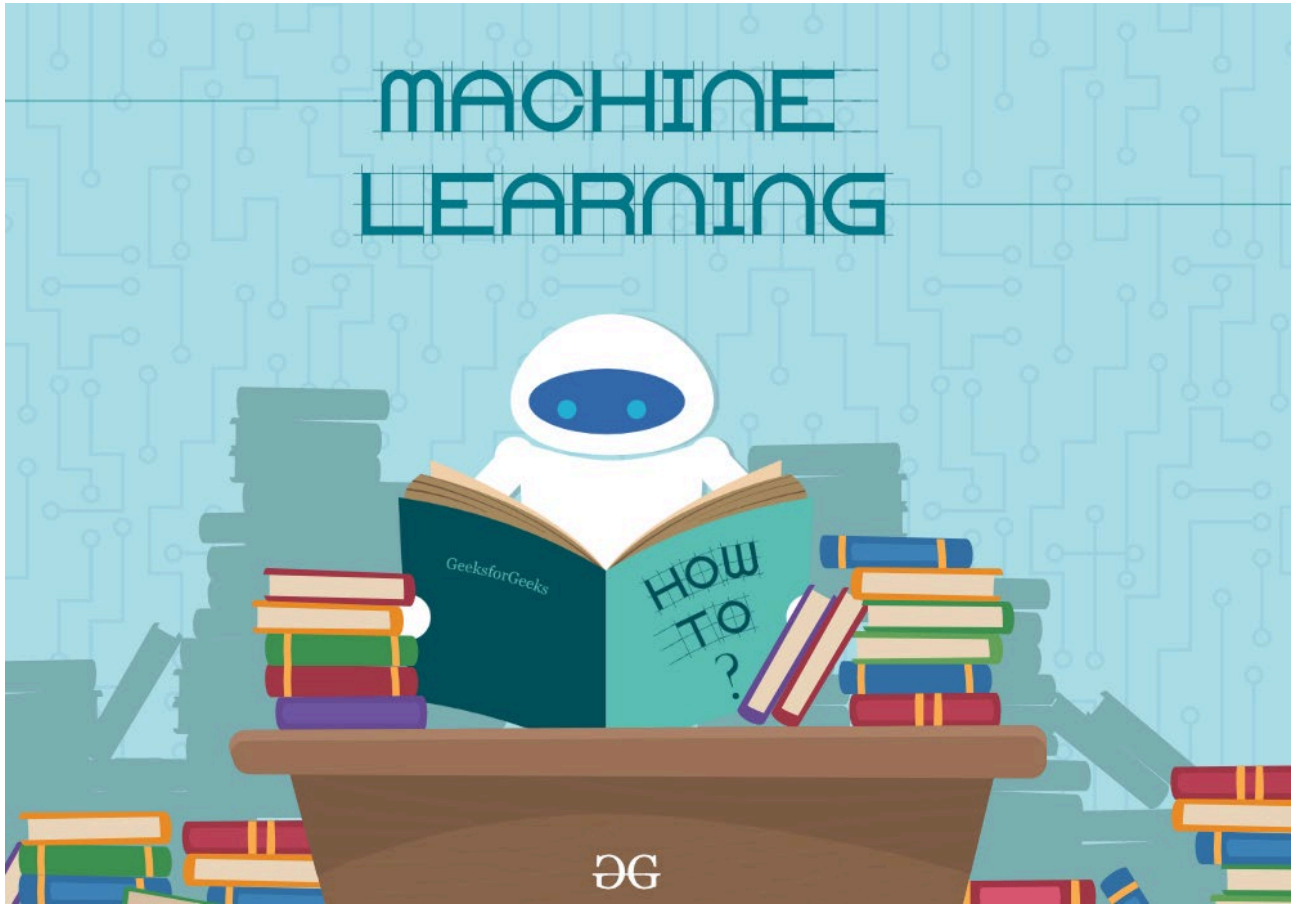
Adam J. Gormley

Associate Professor, Biomedical Engineering, Rutgers University

Outline of Today's Tutorial

1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

So… how does it all work?



Plot twist: You've all done Machine Learning!

A line of best fit is actually ML!

How do you know the difference between a good and bad "Best fit line"?

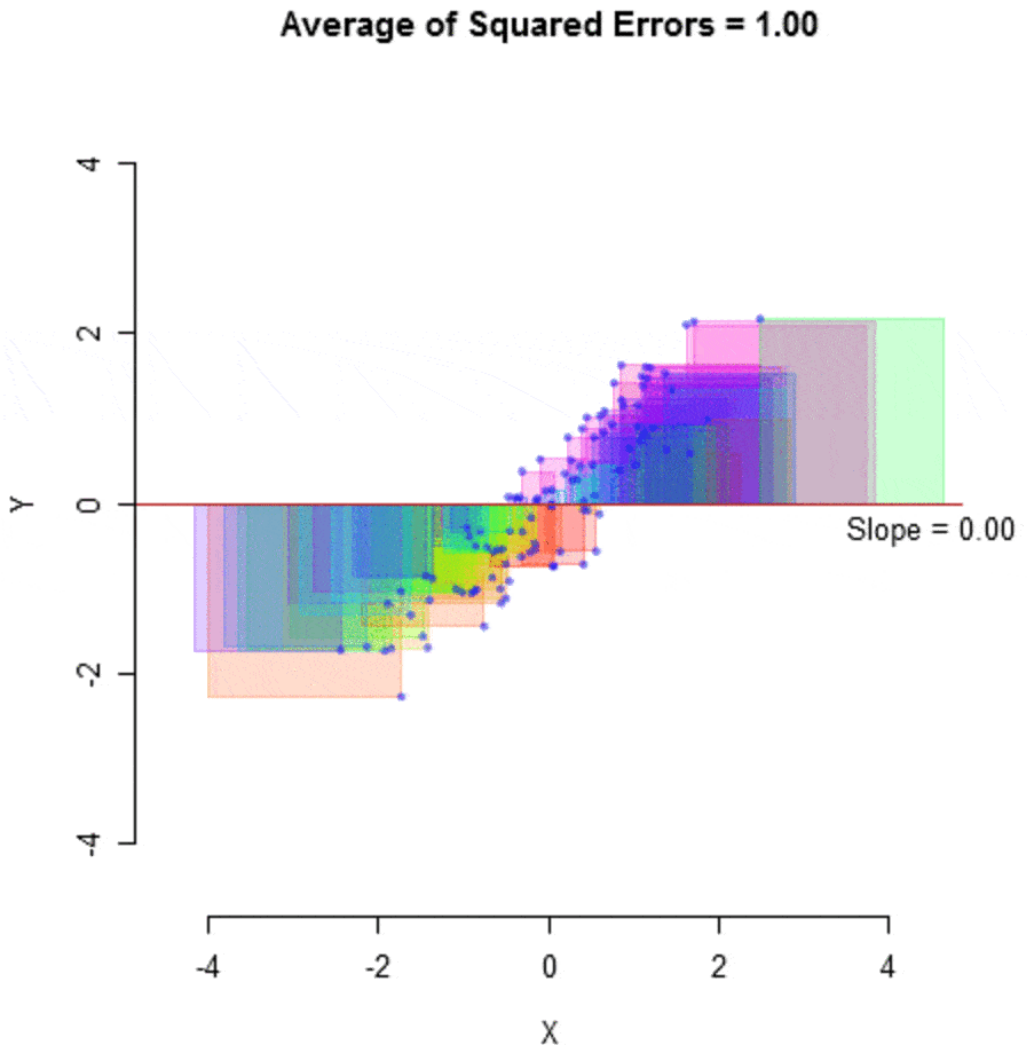
Equation of a line

$$y_i = \alpha + \beta x_i$$

Residual Sum of Squares

$$\mathrm{RSS} = \sum_{i=1}^{n} (\widehat{\varepsilon}_i)^2 = \sum_{i=1}^{n} (y_i - (\widehat{\alpha} + \widehat{\beta}\, x_i))^2$$

Plot twist: You've all done Machine Learning!



How do you know the difference between a good and bad "Best fit line"?

Model

Equation of a line

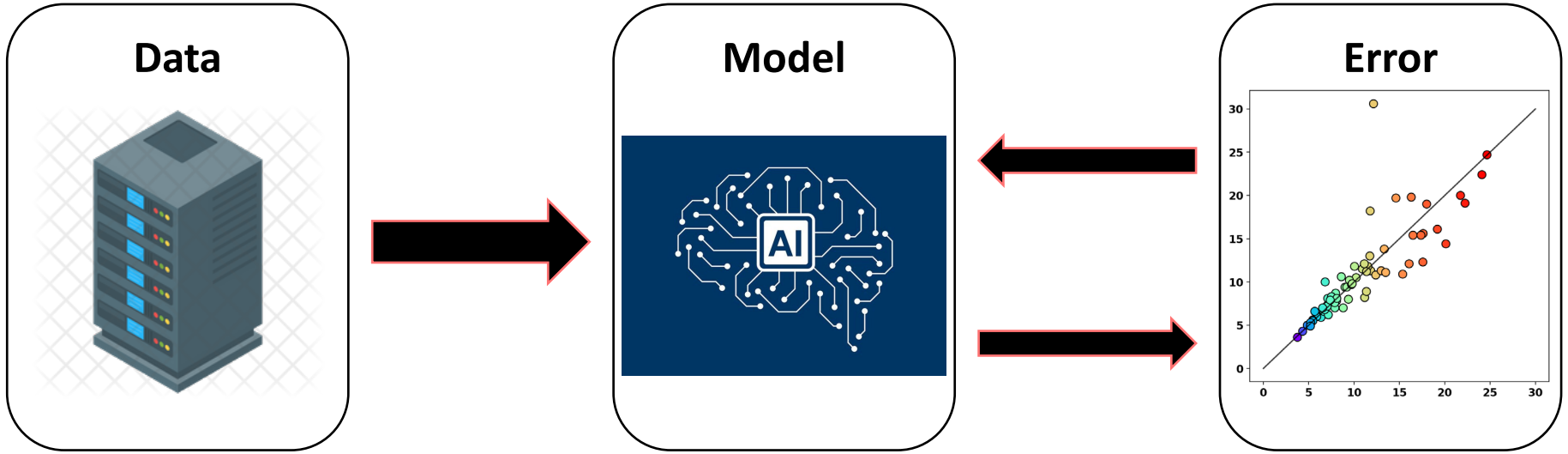
$$y_i = \alpha + \beta x_i$$

Error / Loss

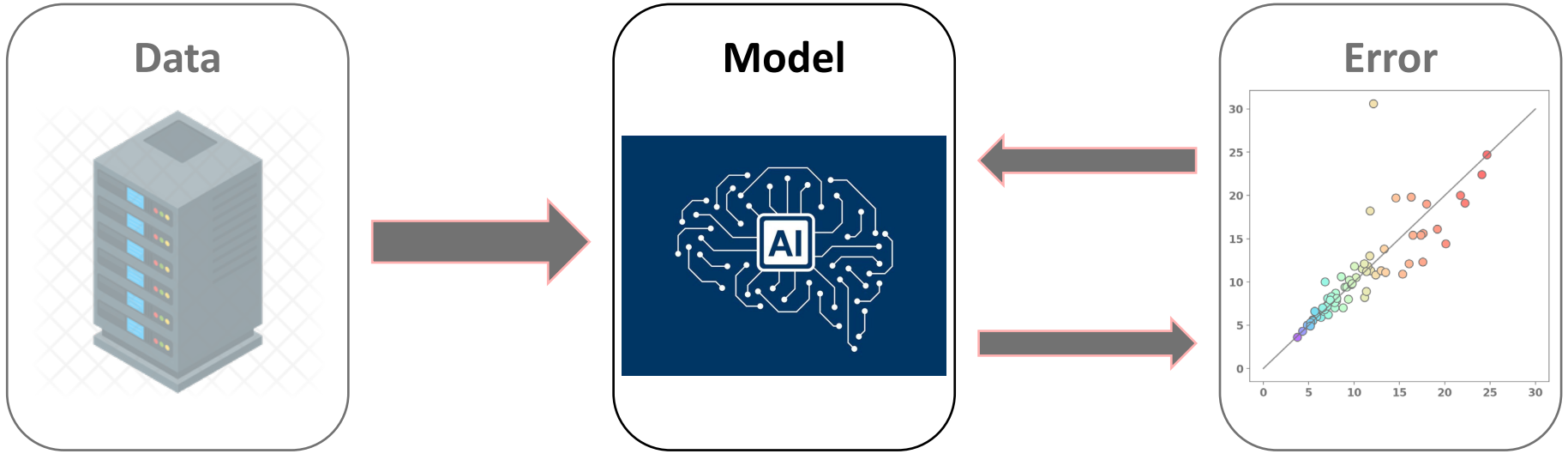
Residual Sum of Squares

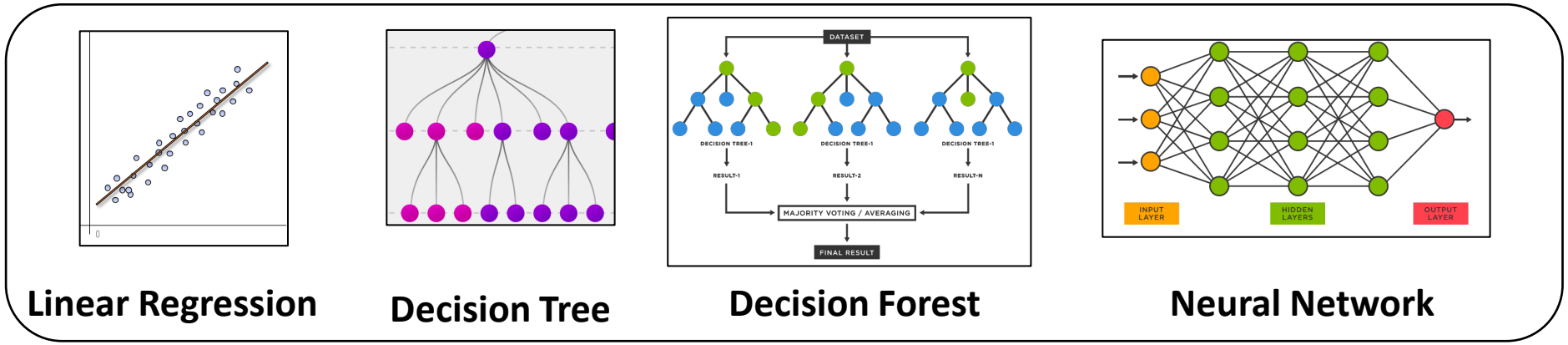
$$\mathrm{RSS} = \sum_{i=1}^{n} (\widehat{\varepsilon}_i)^2 = \sum_{i=1}^{n} (y_i - (\widehat{\alpha} + \widehat{\beta}\, x_i))^2$$

The Universal Framework of Machine Learning



The Universal Framework of Machine Learning





Simple ⟵ ⟶ Complex

ACS POLYMERS Au
AN OPEN ACCESS JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

pubs.acs.org/polymerau

Perspective

A User's Guide to Machine Learning for Polymeric Biomaterials

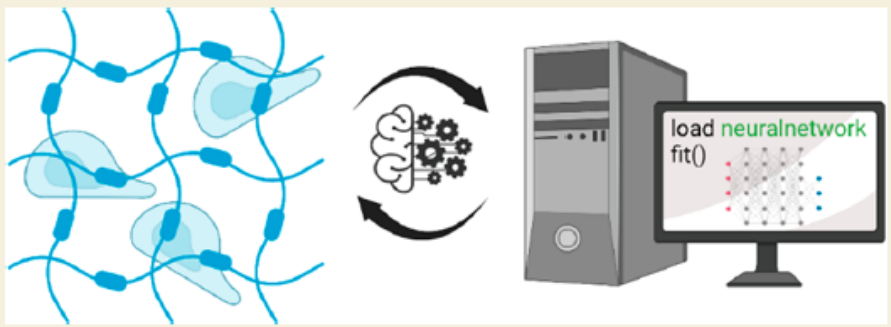
Travis A. Meyer, Cesar Ramirez, Matthew J. Tamasi, and Adam J. Gormley*

Cite This: https://doi.org/10.1021/acspolymersau.2c00037

Read Online

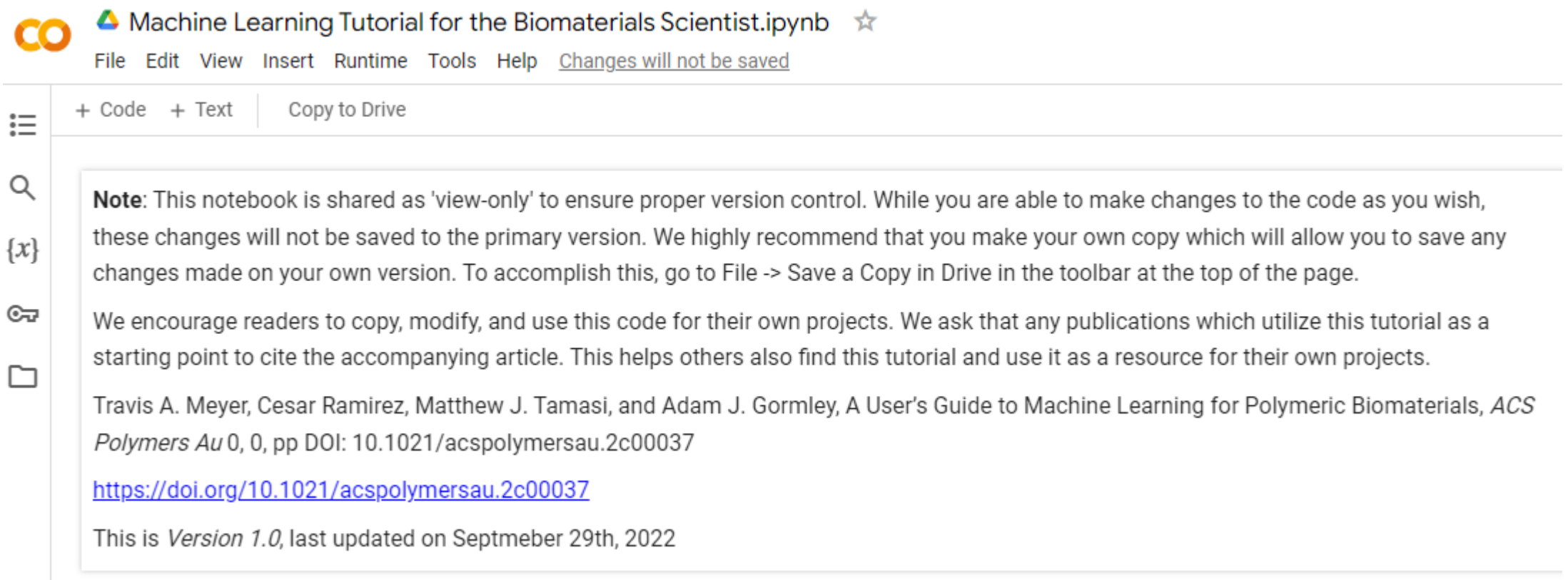
ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: The development of novel biomaterials is a challenging process, complicated by a design space with high dimensionality. Requirements for performance in the complex biological environment lead to difficult *a priori* rational design choices and time-consuming empirical trial-and-error experimentation. Modern data science practices, especially artificial intelligence (AI)/machine learning (ML), offer the promise to help accelerate the identification and testing of next-generation biomaterials. However, it can be a daunting task for biomaterial scientists unfamiliar with modern ML techniques to begin incorporating these useful tools into their development pipeline. This Perspective lays the foundation for a basic understanding of ML while providing a step-by-step guide to new users on how to begin implementing these techniques. A tutorial Python script has been developed walking users through the application of an ML pipeline using data from a real biomaterial design challenge based on group's research. This tutorial provides an opportunity for readers to see and experiment with ML and its syntax in Python. The Google Colab notebook can be easily accessed and copied from the following URL: www.gormleylab.com/MLcolab



KEYWORDS: *Biomaterials, Machine Learning, High throughput, Tutorial, Education and training*

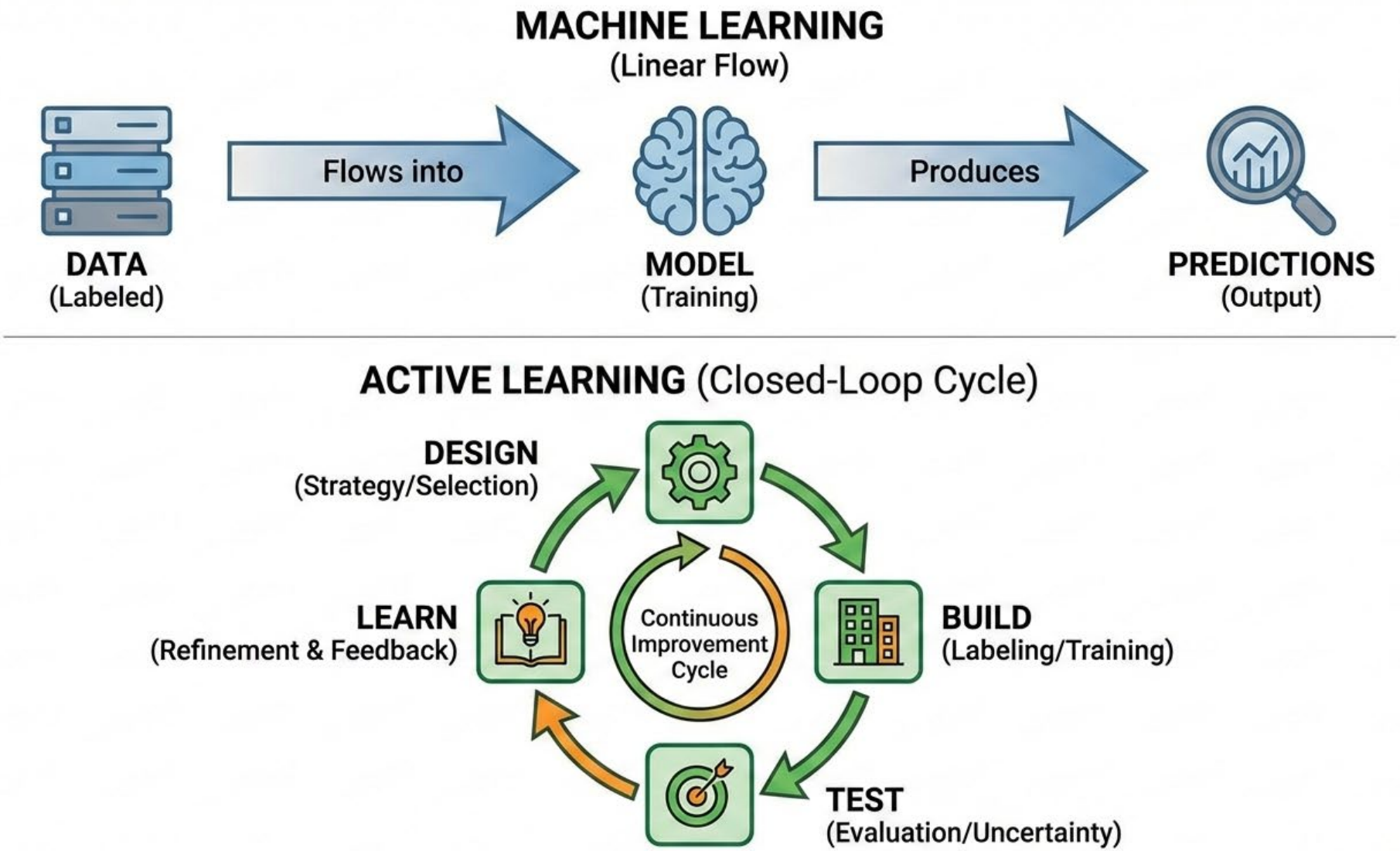
http://www.gormleylab.com/MLcolab



Outline of Today's Tutorial

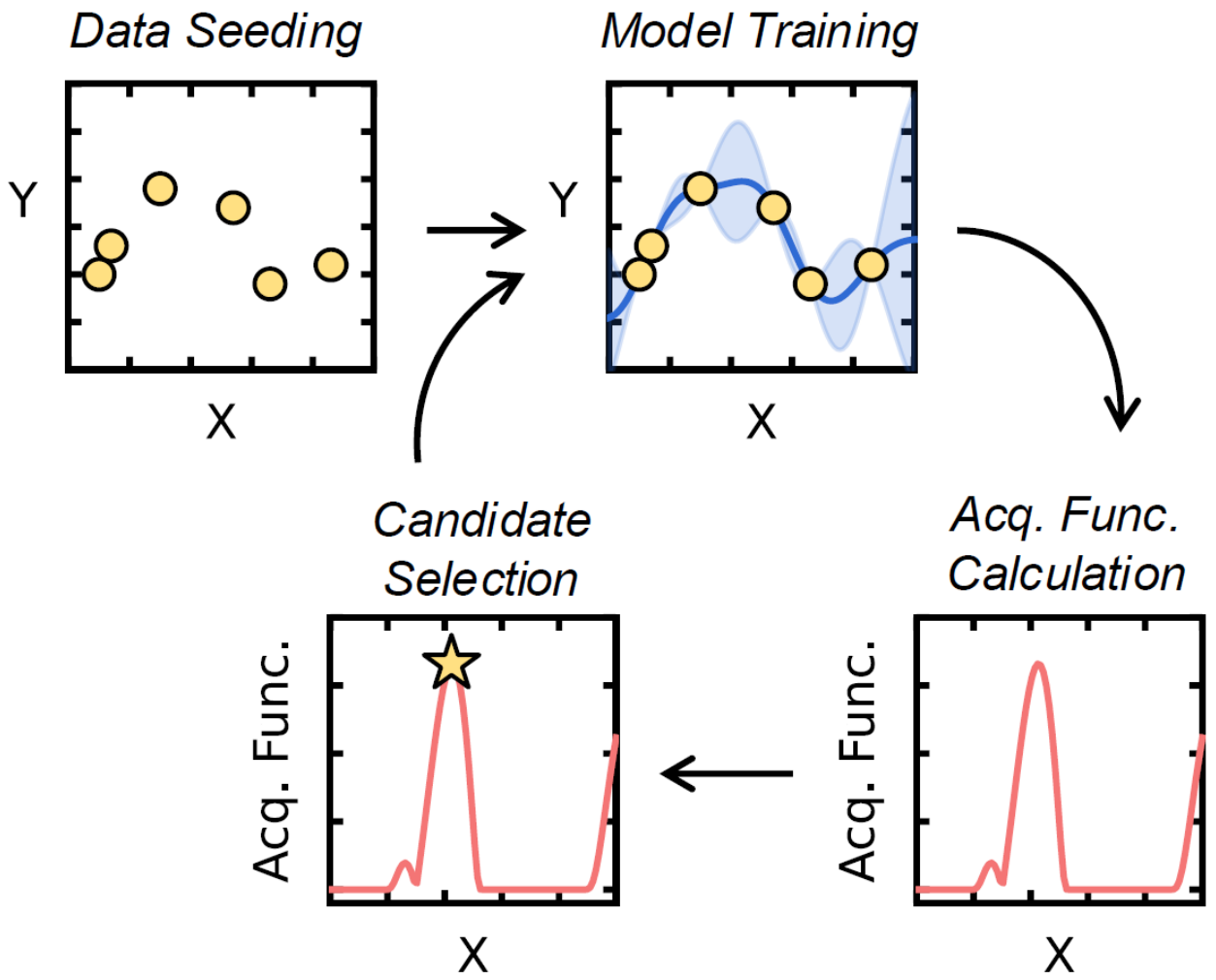
1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

Active Learning vs. Machine Learning



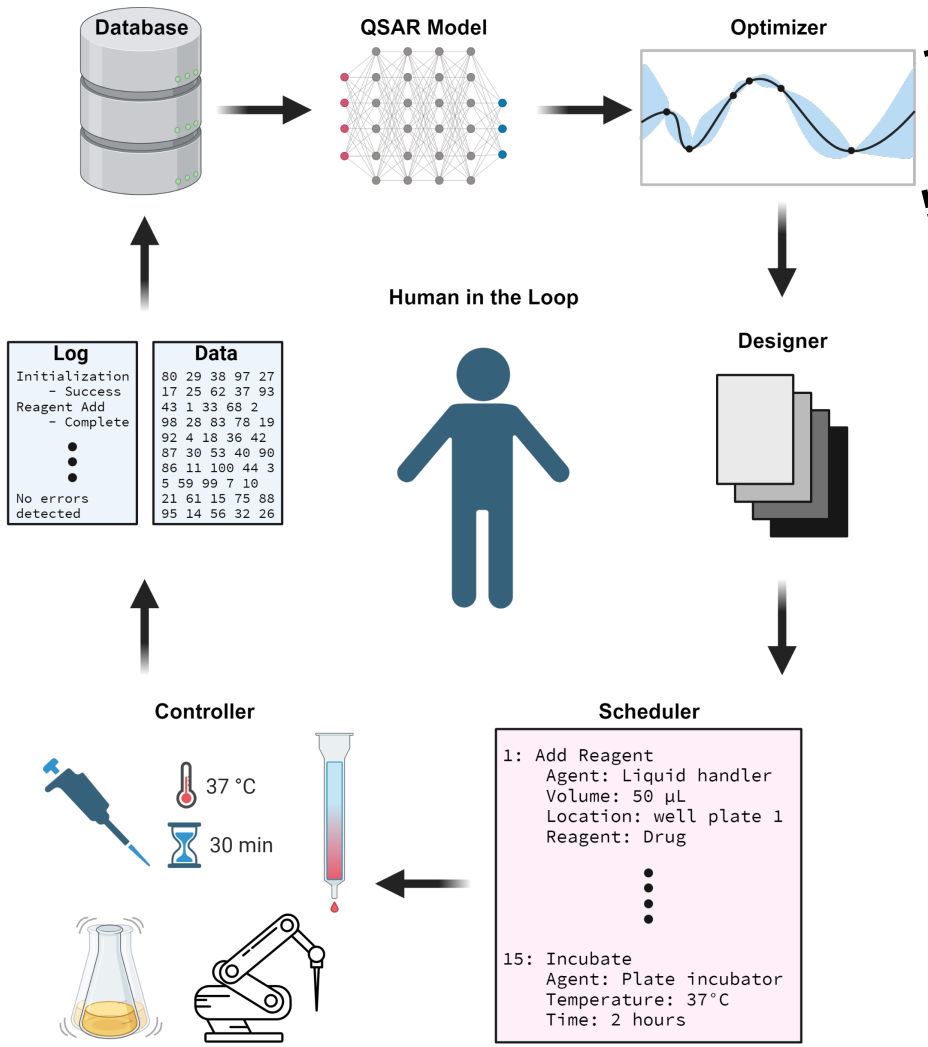
Bayesian Optimization

1. Process starts with data seeding
   - Many approaches, we use Latin hypercube sampling (LHS)
2. Train a model that outputs uncertainty
   - Gaussian process regressor (GPR) is most common
3. Maximize the acquisition function
   - Probability you will learn the most information from a new data point
   - Expected improvement (EI) is most common
4. Measure data point to learn ground truth
5. Rinse and repeat

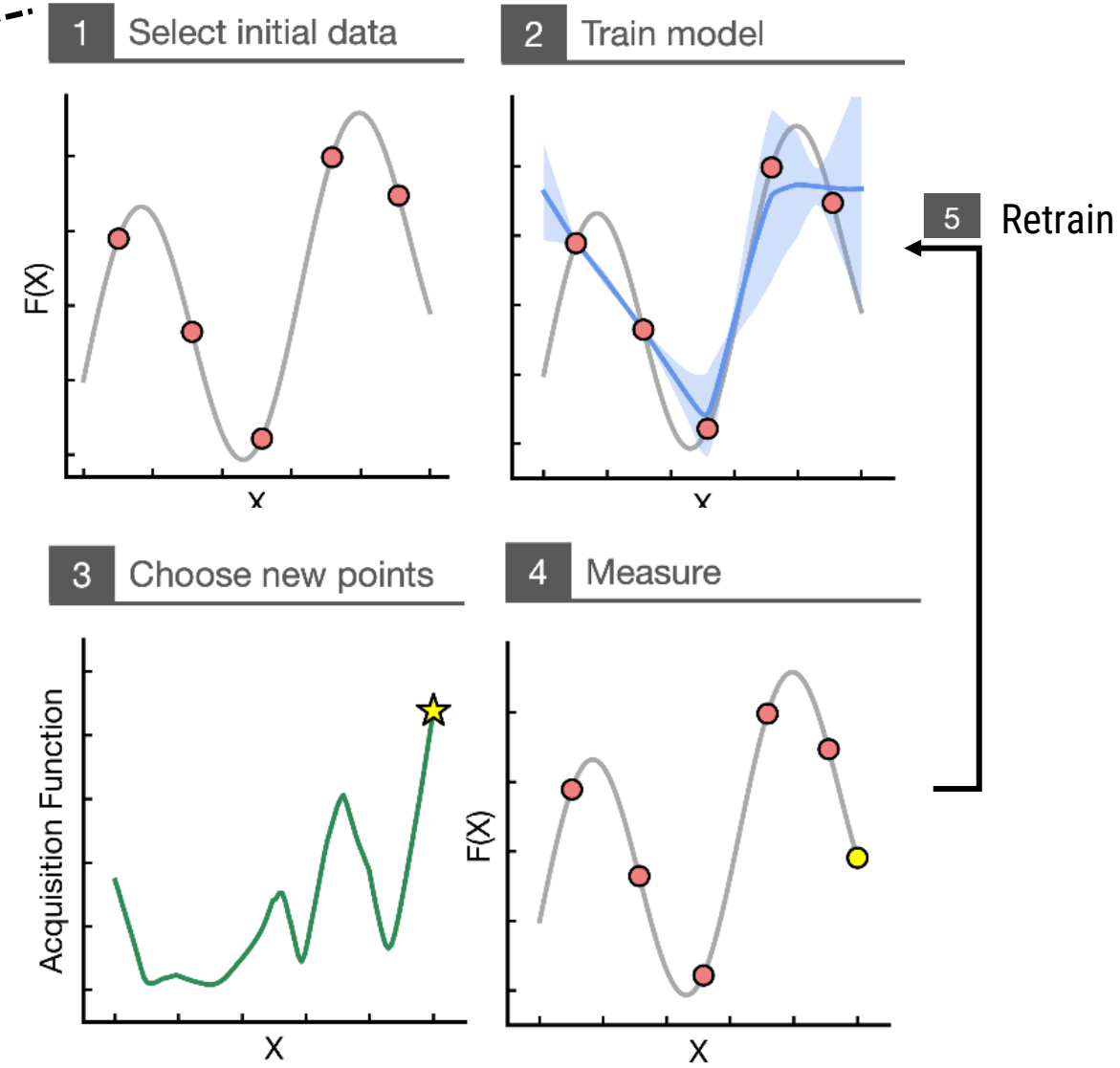


A. Maroulis, et al. A user's guide to your first self-driving liquid handling lab. In Revision at Digital Discovery. https://chemrxiv.org/doi/full/10.26434/chemrxiv-2025-6ws9m

Active Learning and Self-Driving Labs



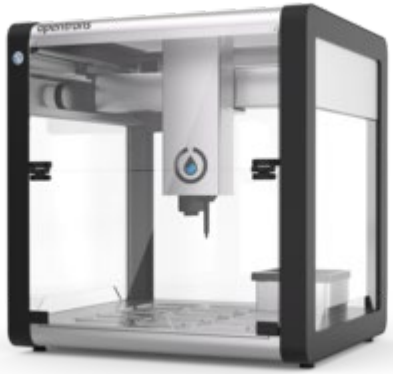
Active Learning

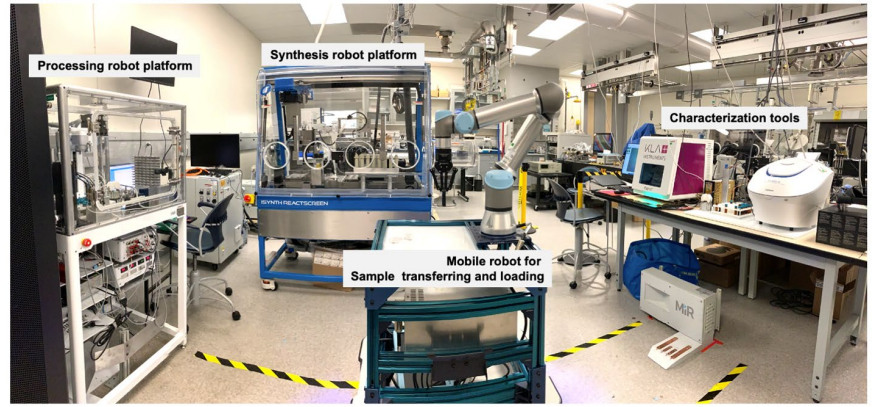


Outline of Today's Tutorial

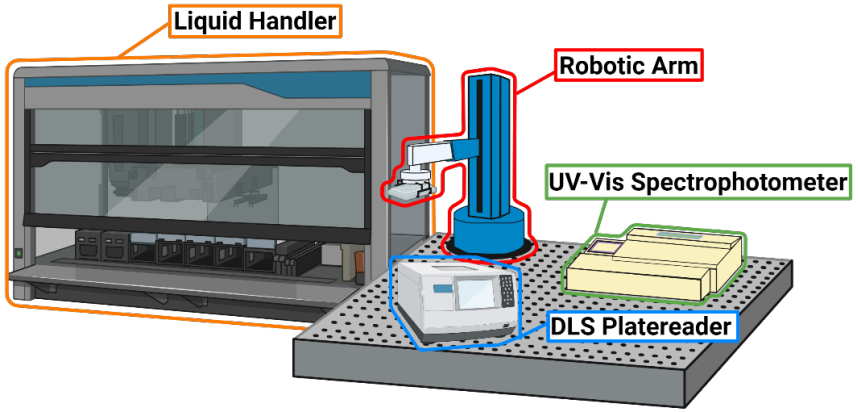
1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

Democratizing Automation





?



Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2026, **5**, 2028

A user's guide to your first self-driving liquid handling lab

Apostolos P. Maroulis,†[a] Dylan M. Waynor,†[a] Quinn M. Gallagher,[b] Roshan A. Patel,[b] Matthew Tamasi,[a] D. Christopher Radford,[a] Michael A. Webb *[b] and Adam J. Gormley *[a]

Experimentation is inherently difficult because most methods require substantial refinement, calibration, and validation before high-quality, reliable data can be collected. In most cases, experimental outcomes are impacted by multiple variables, thus requiring their simultaneous optimization for single and multi-objective targets. Traditional experimental approaches rely on trial-and-error methods guided by rational decision making, but these become increasingly inefficient and ineffective as complex interactions between inputs limit our ability to capture underlying trends using conventional statistical approaches. Machine learning and active learning (ML/AL) combined with automation represents an approach that can bolster future laboratory productivity. However, a steep initial learning curve and high costs of instrumentation pose substantial barriers to adoption. To democratize access, we herein comprehensively cover both the computational skills and hardware implementation necessary for self-driven experimental workflows. The accompanying open-source, low-cost liquid handling platforms offer practical templates for researchers adopting self-driving lab (SDL) methodologies. Complete tutorials and build guides are provided at https://gormleylab.github.io/SDLGuide.

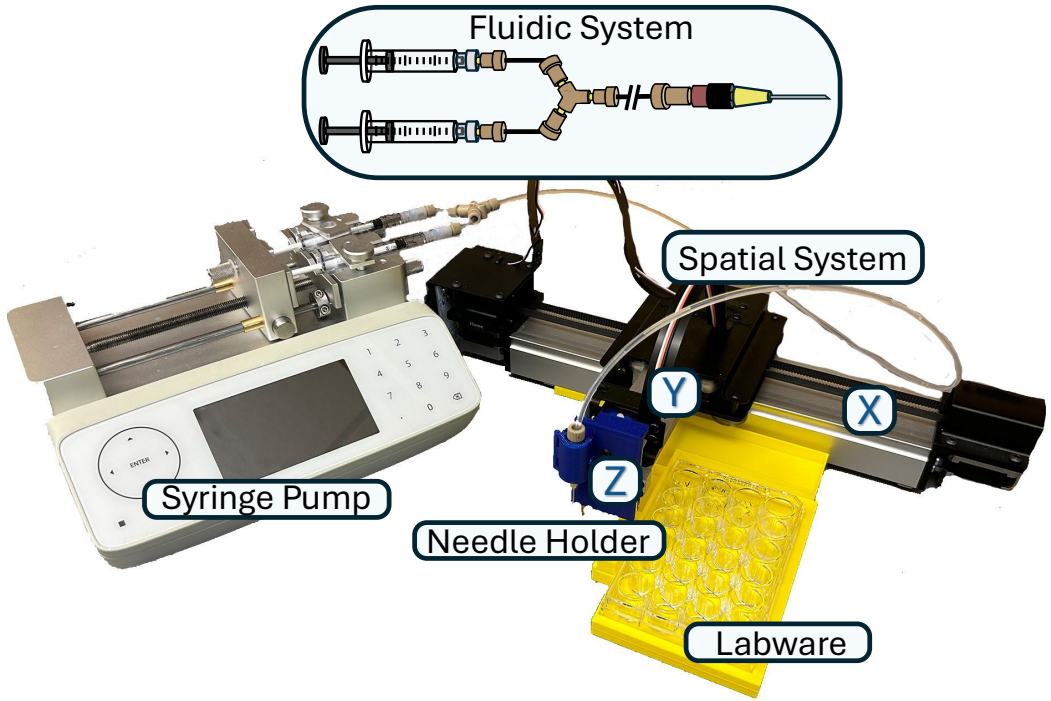
Received 25th November 2025
Accepted 25th March 2026

DOI: 10.1039/d5dd00525f

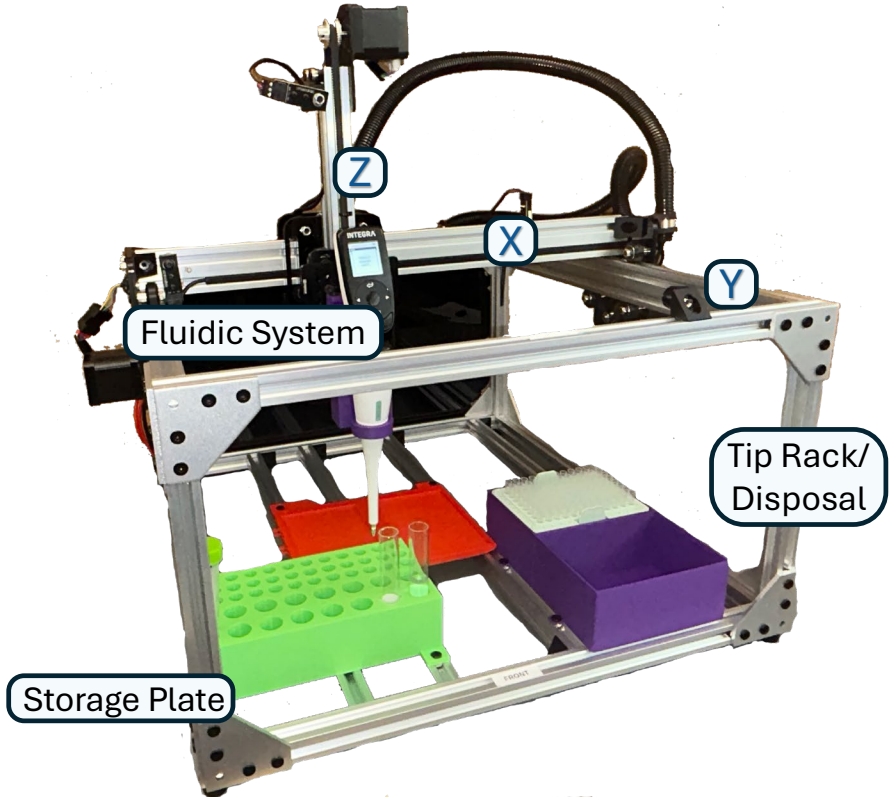
rsc.li/digitaldiscovery

DIY Liquid Handlers

Pen Plotter Driven Liquid Handler

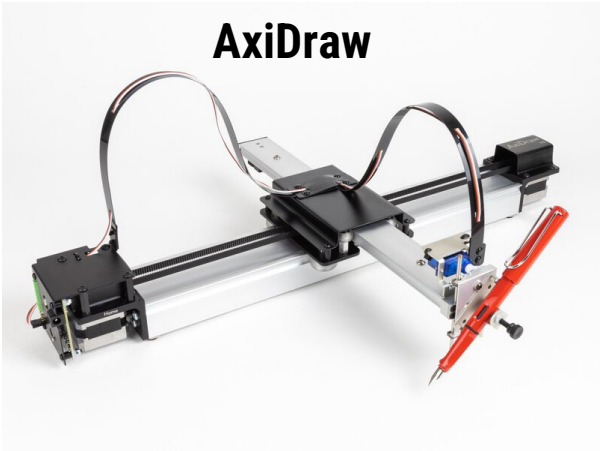


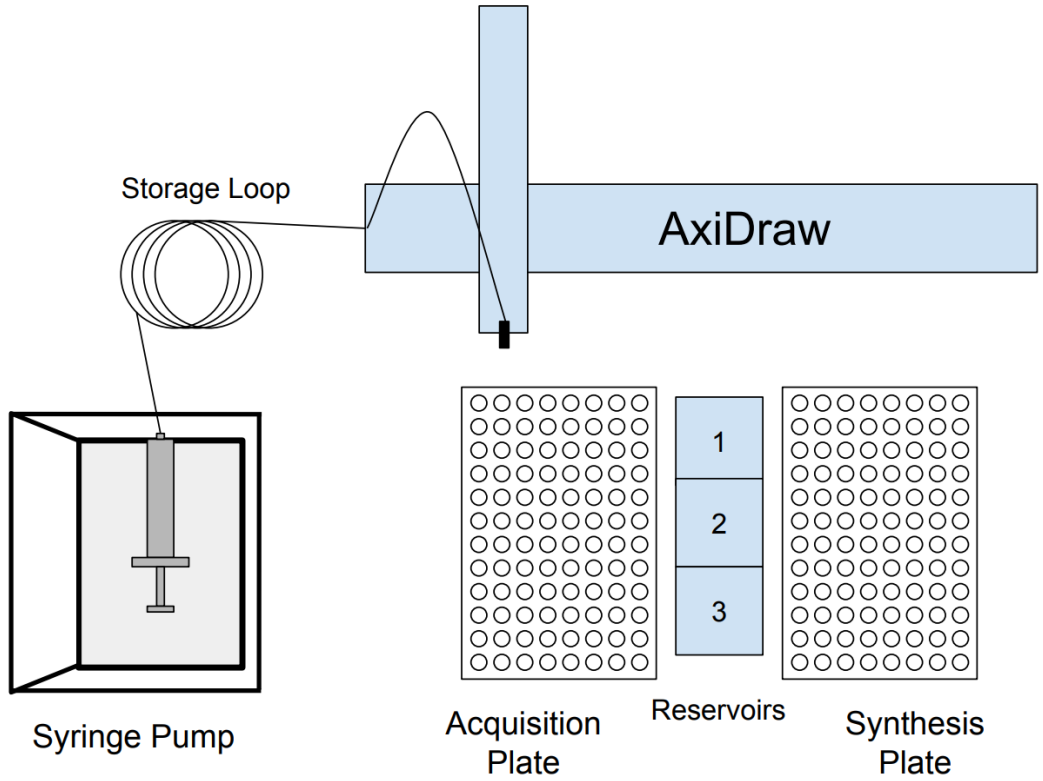
Pipette Driven Liquid Handler

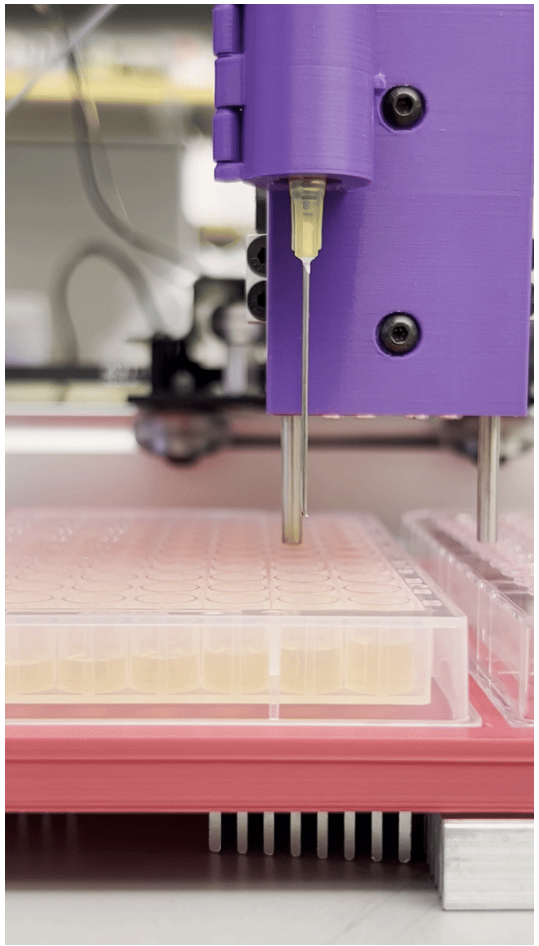


< $1,000 each excluding the e-pipette and syringe pump

Minimum Example of a Self-Driving Liquid Handling Lab







A. Maroulis, et al. A user's guide to your first self-driving liquid handling lab. In Revision at Digital Discovery. https://chemrxiv.org/doi/full/10.26434/chemrxiv-2025-6ws9m

Welcome to the User's Guide to AL/SDLs Repository

Explore build guides, software, and hands-on tutorials for automated liquid handling and experiment optimization.

View on GitHub

Liquid Handler Platforms

Choose your platform to access build guides, software, and documentation.

Pen Plotter Liquid Handler

Original AxiDraw-based liquid handler with build guides and SOP.

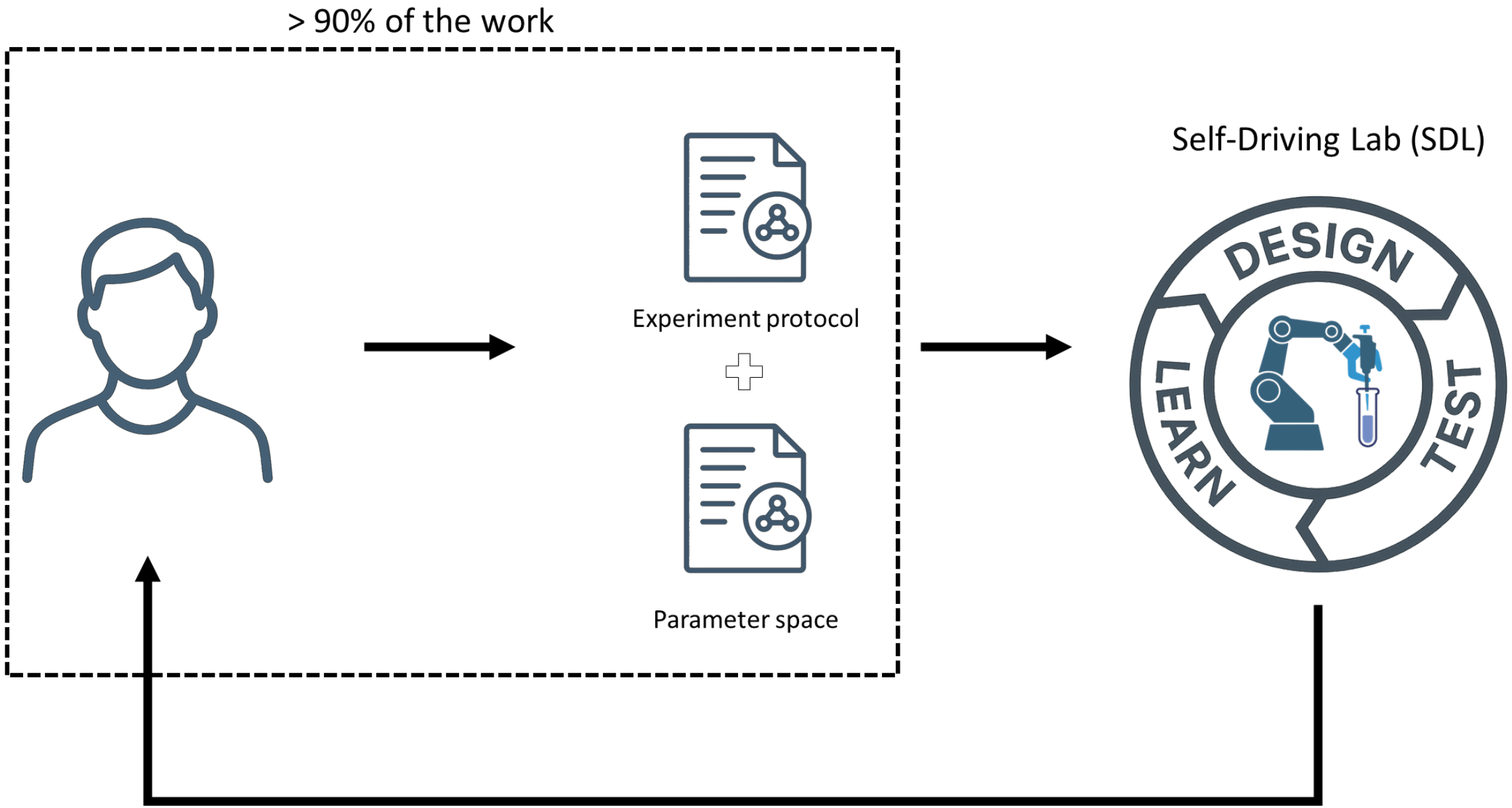
- Pen Plotter Github
- Build Guide
- Software SOP
- Software Download

Pipette Liquid Handler

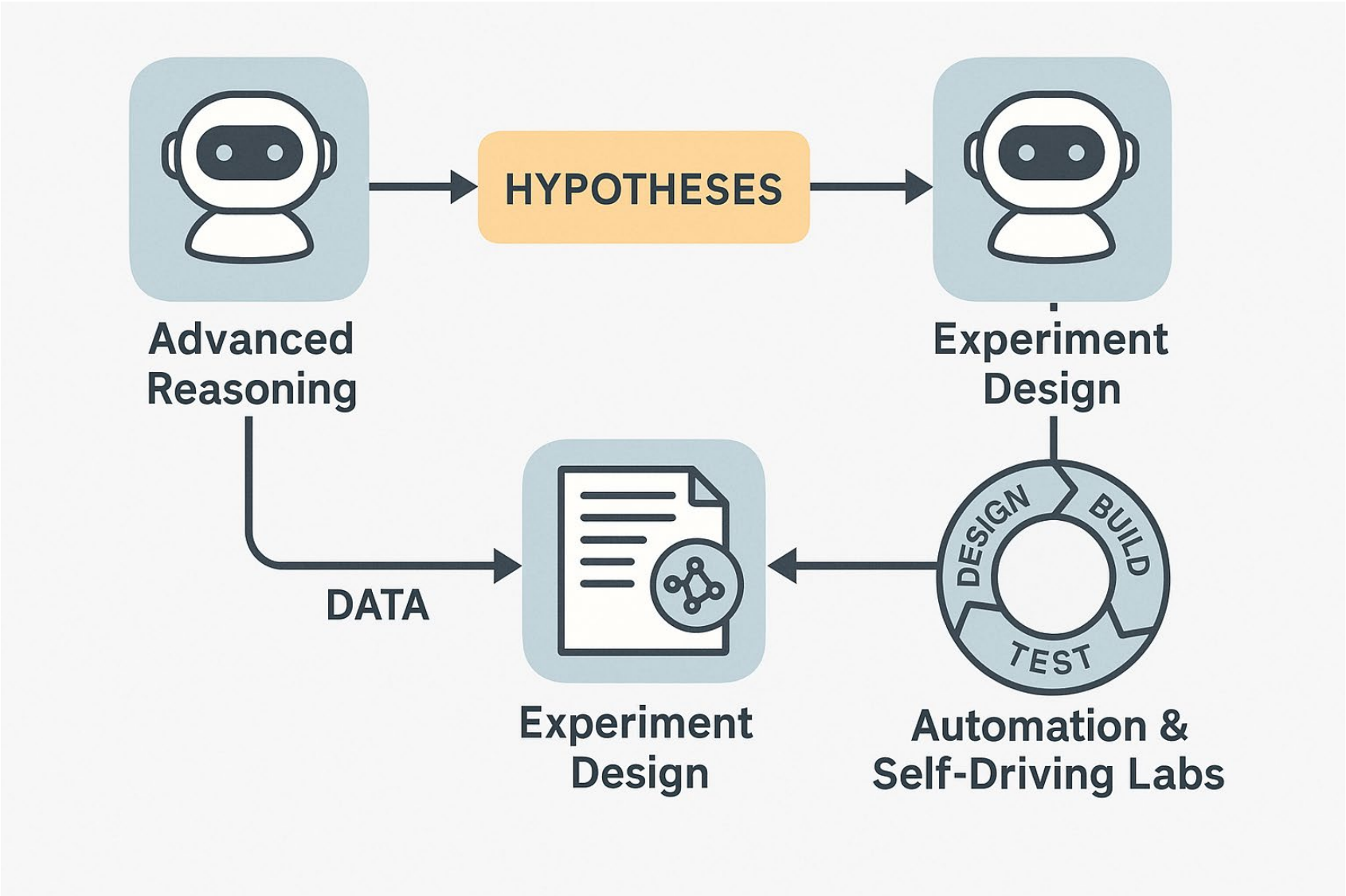
Automated pipetting platform with SDL software and full build documentation.

- Pipette GitHub
- Build Guide
- Parts List
- Software Download

BIG Problem with Self-Driving Labs



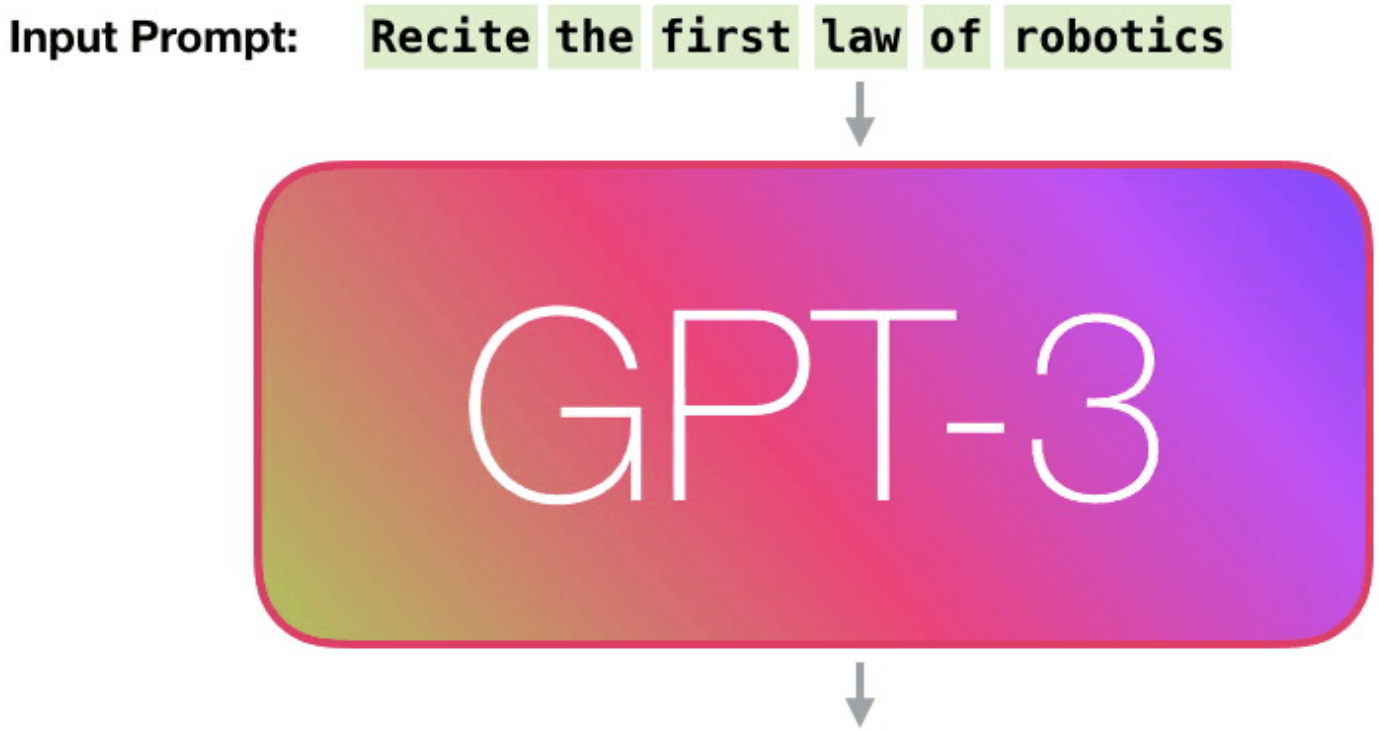
The Rise of AI Agents within the Scientific Method



Outline of Today's Tutorial

1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

Large Language Models



Next-token prediction

Transformer Architectures

Attention Is All You Need

2017

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

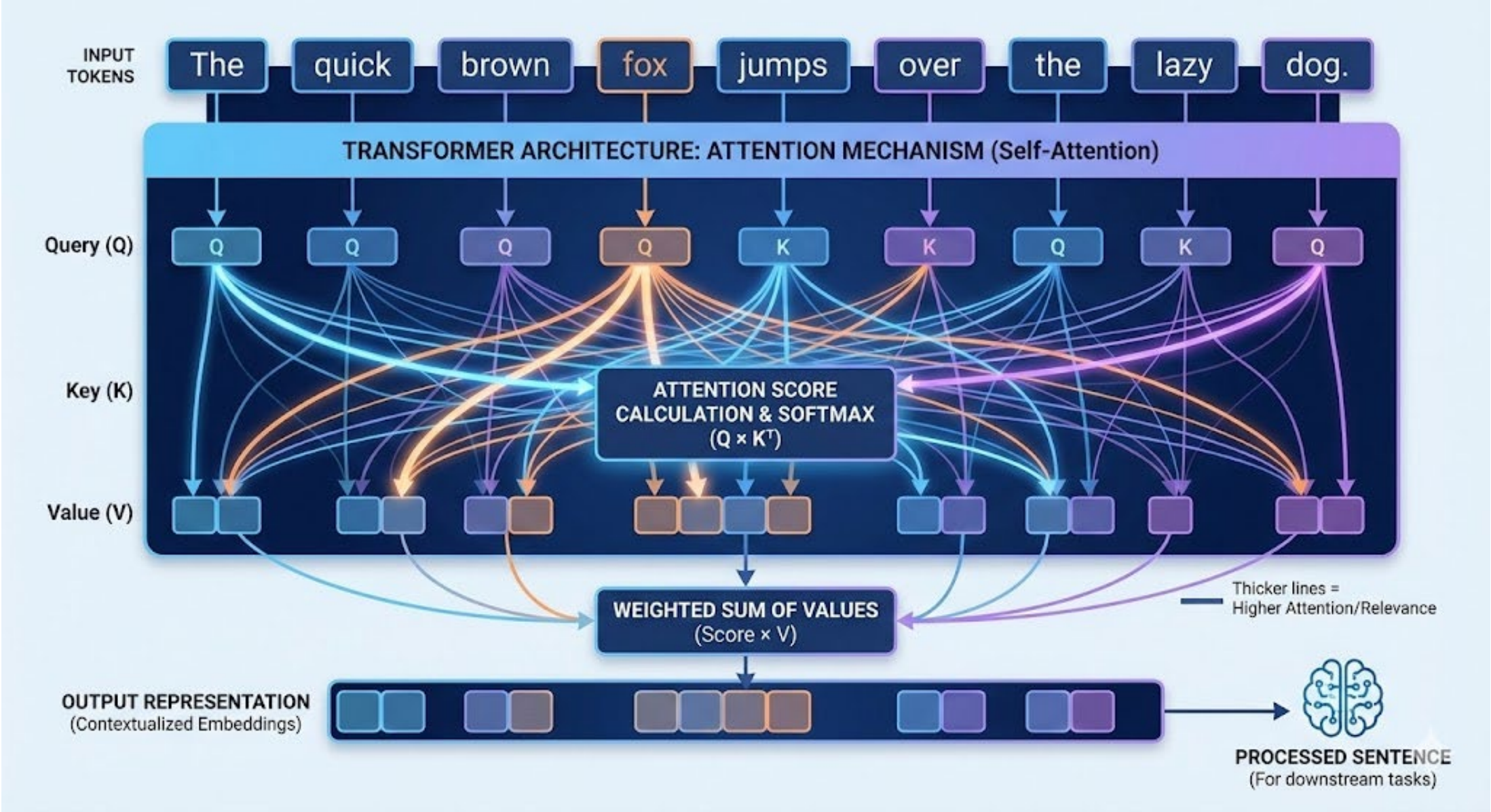
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best

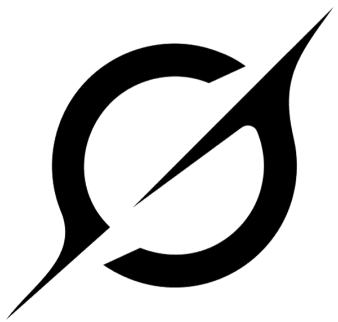
Transformer Architectures



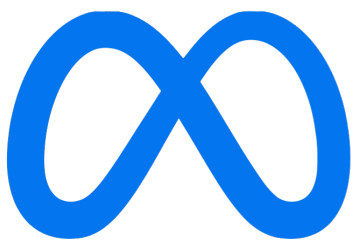
Foundation Models

ChatGPT

Claude



Grok



LLaMA

Calling LLMs in Python

Key Components

- Model
  - Claude: Sonnet, Opus, Haiku
- System prompt
  - ‘You are a drug delivery scientist’
- User prompt
  - ‘What polymer is most common in microparticles?’

```python
import anthropic

client = anthropic.Anthropic()

message = client.messages.create(
    model="claude-sonnet-4-5",
    messages=[
        {
            "role": "user",
            "content": "How is this presentation going?"
        }
    ]
)
print(message.content)
```

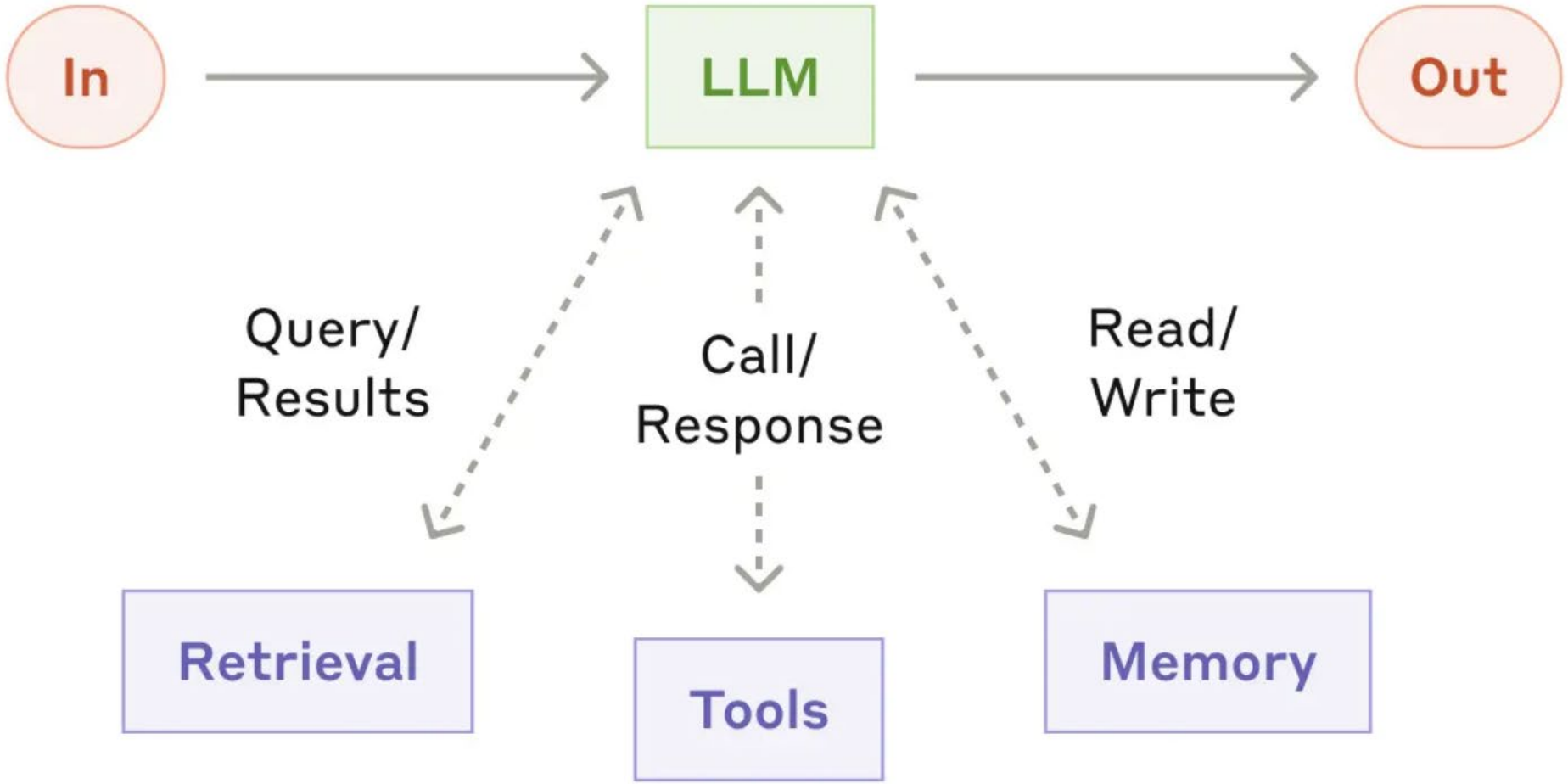
```
text="You're not as funny as you think you are"
```

Outline of Today's Tutorial

1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

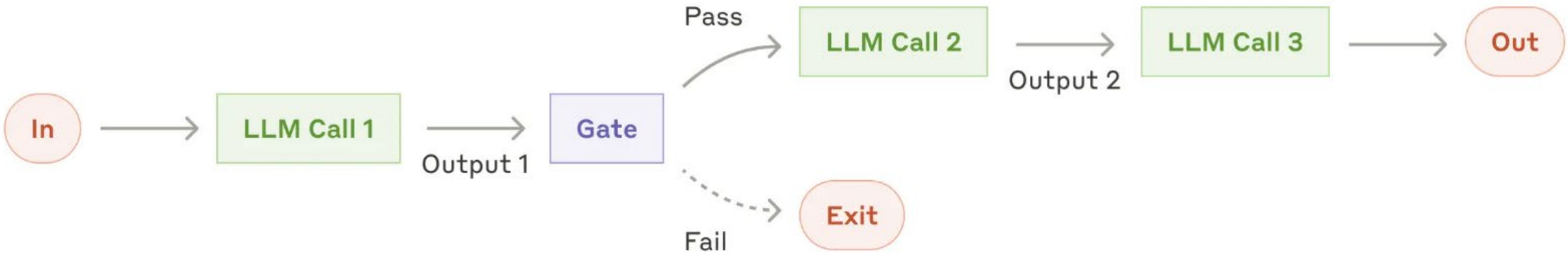
AI Agents

ANTHROP\C



Agent Workflows

ANTHROP\C



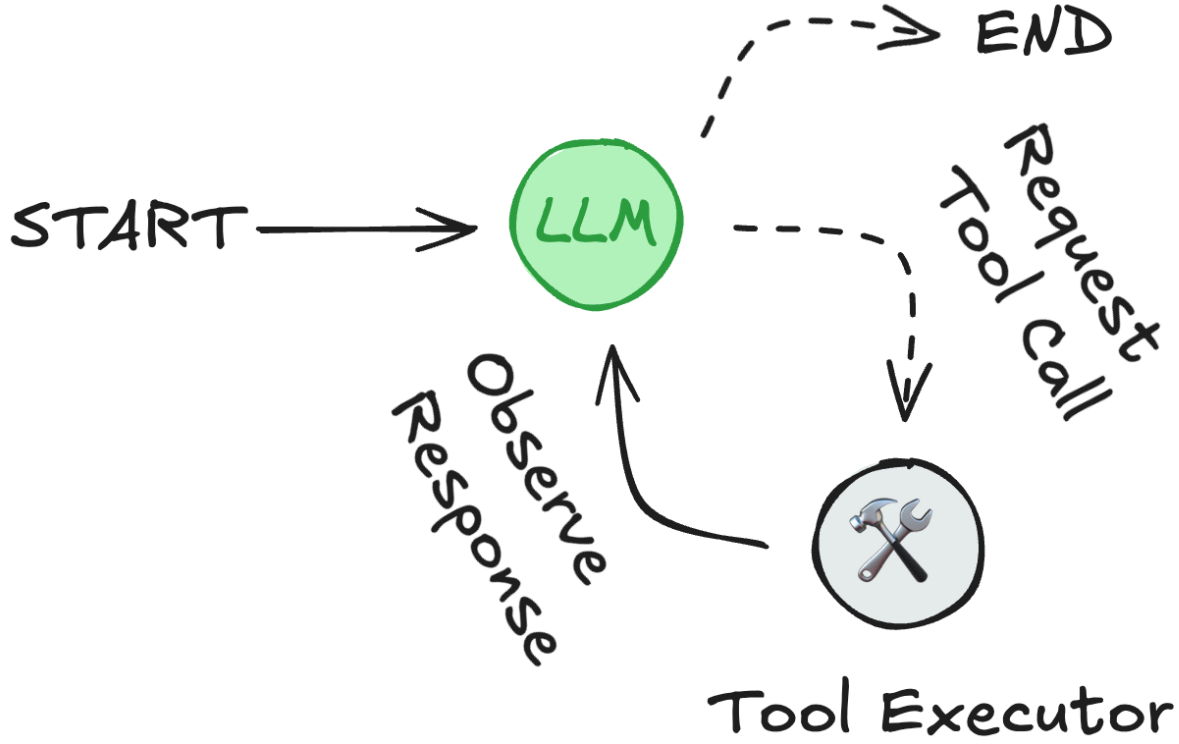
Outline of Today's Tutorial

1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

The Infamous Tool Call

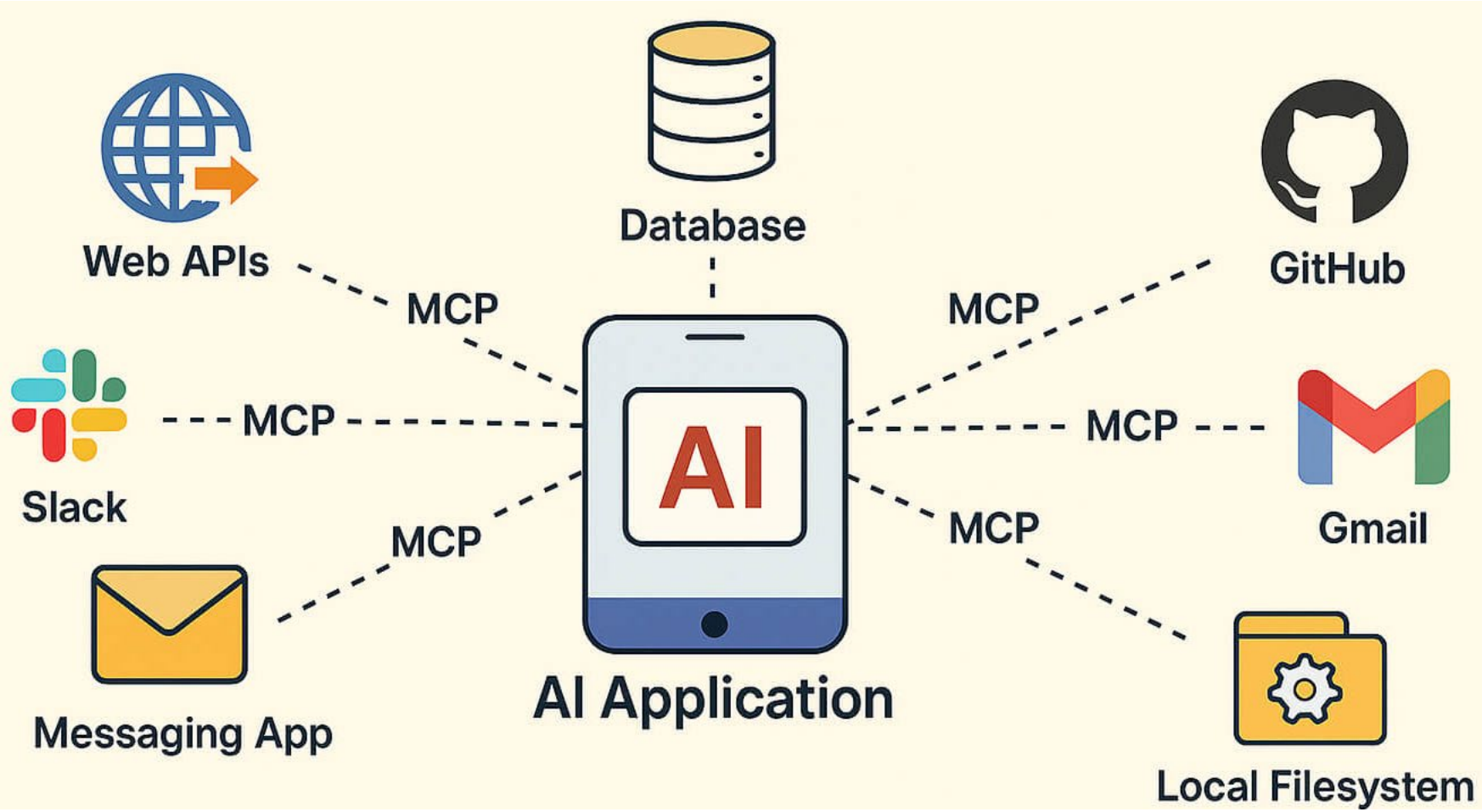
Tool Call Loop

1. Put an LLM inside a while loop
2. Provide them with access to the outside world to perform actions
   - Tool calls
3. Keep looping until the LLM stops calling new tools and is done



Model Context Protocol (MCP)

ANTHROP\C



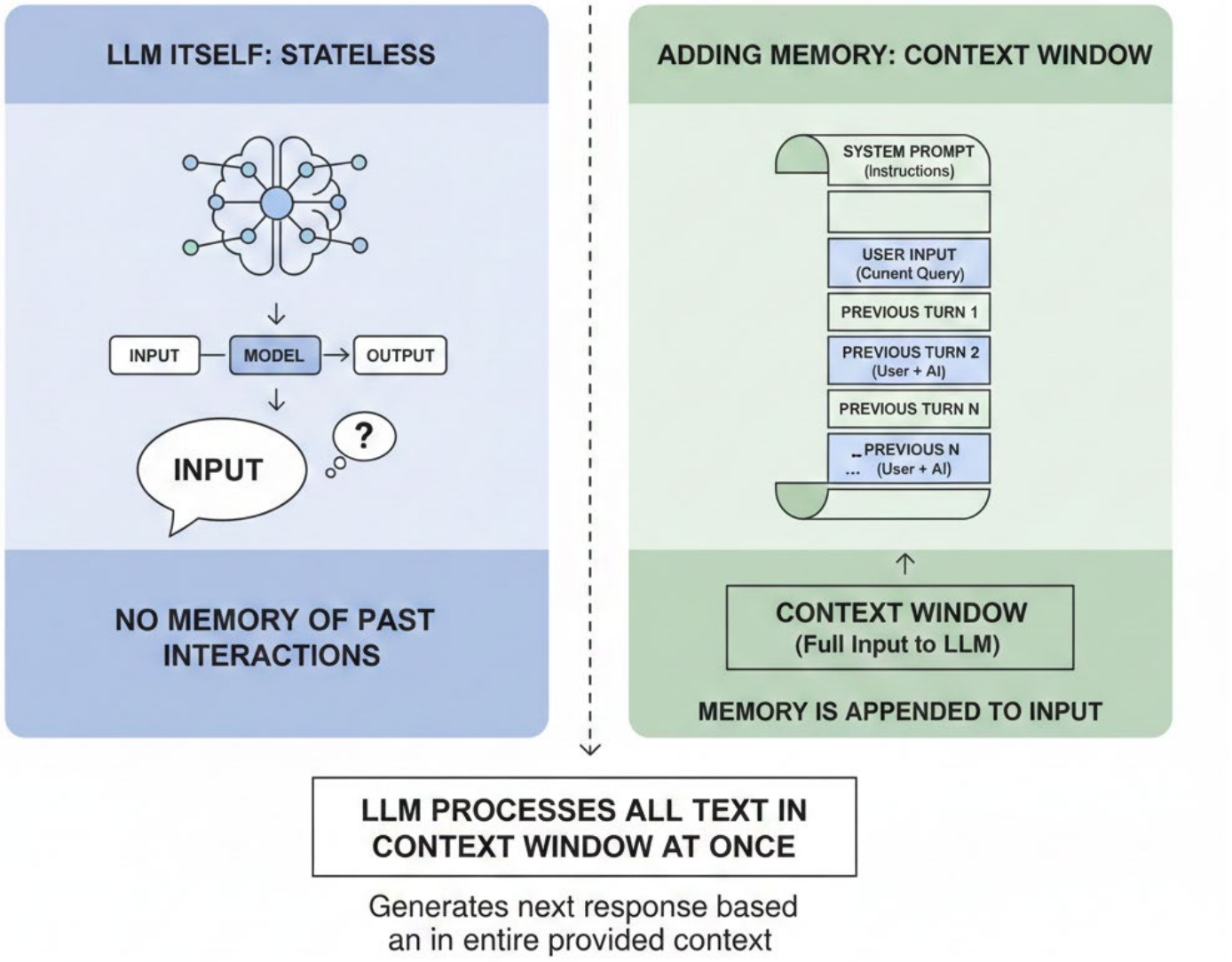
Standardized protocol for sharing tools

Outline of Today's Tutorial

1. Machine learning
2. Active learning
3. Self-driving labs
4. Open-source automation
5. Large language models
6. Calling LLMs in Python
7. AI Agents
9. Agentic tool calling
10. Model context protocol (MCP)
11. Memory, state, and context
12. Retrieval-augmented generation (RAG)

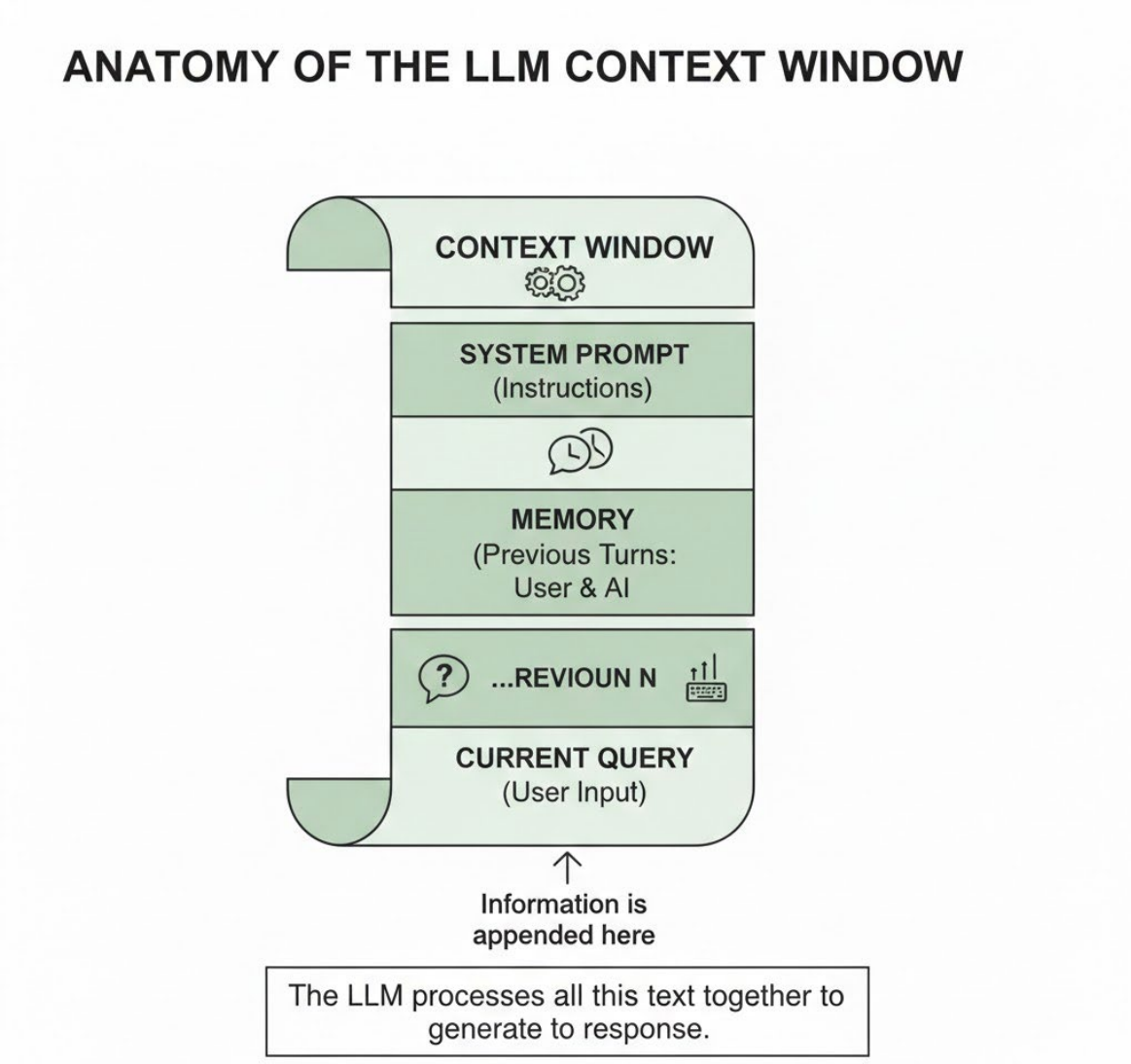
LLMs are Stateless – No Memory

- When you send a prompt to an LLM, it has no intrinsic memory of you or past interactions
- To add memory, you must append previous interactions to the *Context Window*
- Every time you submit a query, you are also sending all of your previous interactions within a session



Context Engineering

- The state (i.e., context window) contains all information being sent to the LLM
- Often 1M tokens or 1,500 pages of text!
- HOWEVER!
  - Too much information leads to context rot
- Context engineering is the process of manicuring quality context



Retrieval-Augmented Generation (RAG)

- Pipeline for retrieving quality knowledge from databases
  - Semantic search of vector databases
- Excellent for searching research papers for context!
- When provided as an MCP server, the LLM can search the database to ‘learn’ new information in milliseconds!