



中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

AI partner at BESIII: DrSai

Part I: Introduction - ML and LLMs&Agents

Part II: Construction of DrSai, application at BESIII

Ke Li (like@ihep.ac.cn)

Part I

Introductions – ML and LLMs & Agents

Purpose: know what it can do and what it can not do !

- There are many many lectures online
 - 李宏毅: <https://www.bilibili.com/video/BV1YsqSY8EiW/>
 - 吴恩达: https://www.bilibili.com/video/BV1Bq421A74G/?spm_id_from=333.337.search-card.all.click
- Or just ask LLM: deepseek, kimi, etc.

人工智能与高能物理的历史

高能物理领域在约40年前已开始应用ML/DL，
如欧洲核子中心的LEP和LHC

助力了多个重要科学发现，如希格斯粒子

ANNs improved the sensitivity of searches for the Higgs boson at the CERN Large Electron-Positron (LEP) collider during the 1990s [44], and were used in the analysis of data that led to its discovery at the CERN Large Hadron Collider in 2012 [45]. ANN were also used in studies of the top quark at Fermilab [46].

[2024 The Nobel Committee for Physics](#)

Identification of tau decays using a neural network

V. Innocente (CERN and INFN, Naples), Y.F. Wang (INFN, Florence and Florence U.), Z.P. Zhang (World Lab, Lausanne and Hefei, CUST)

Jun, 1992

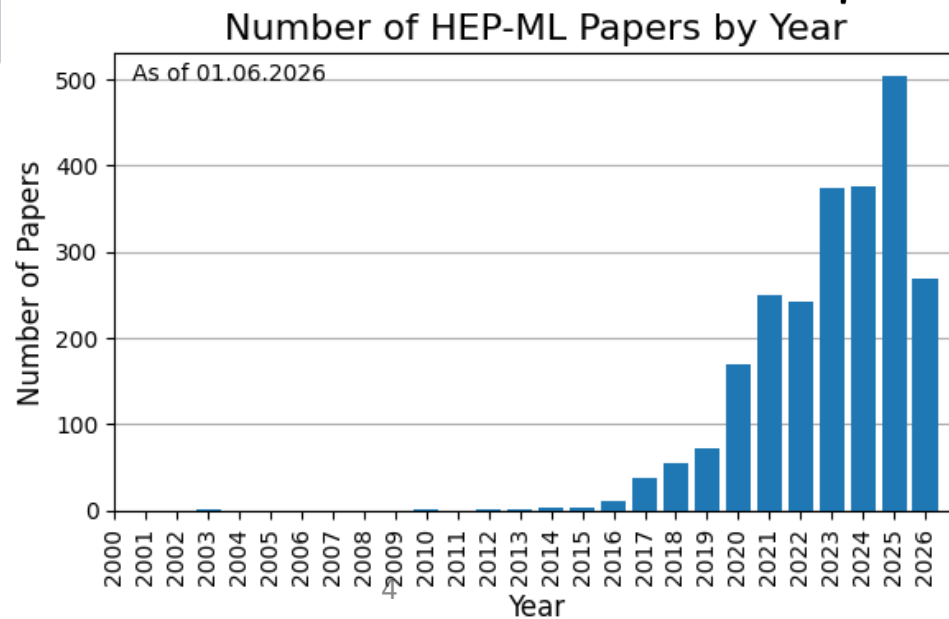
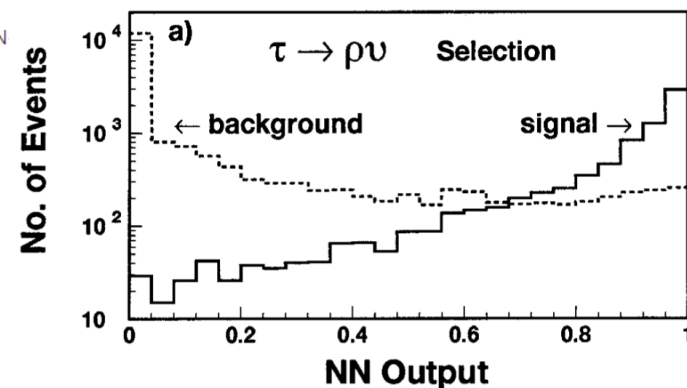
28 pages

Published in: *Nucl.Instrum.Meth.A* 323 (1992) 647-656

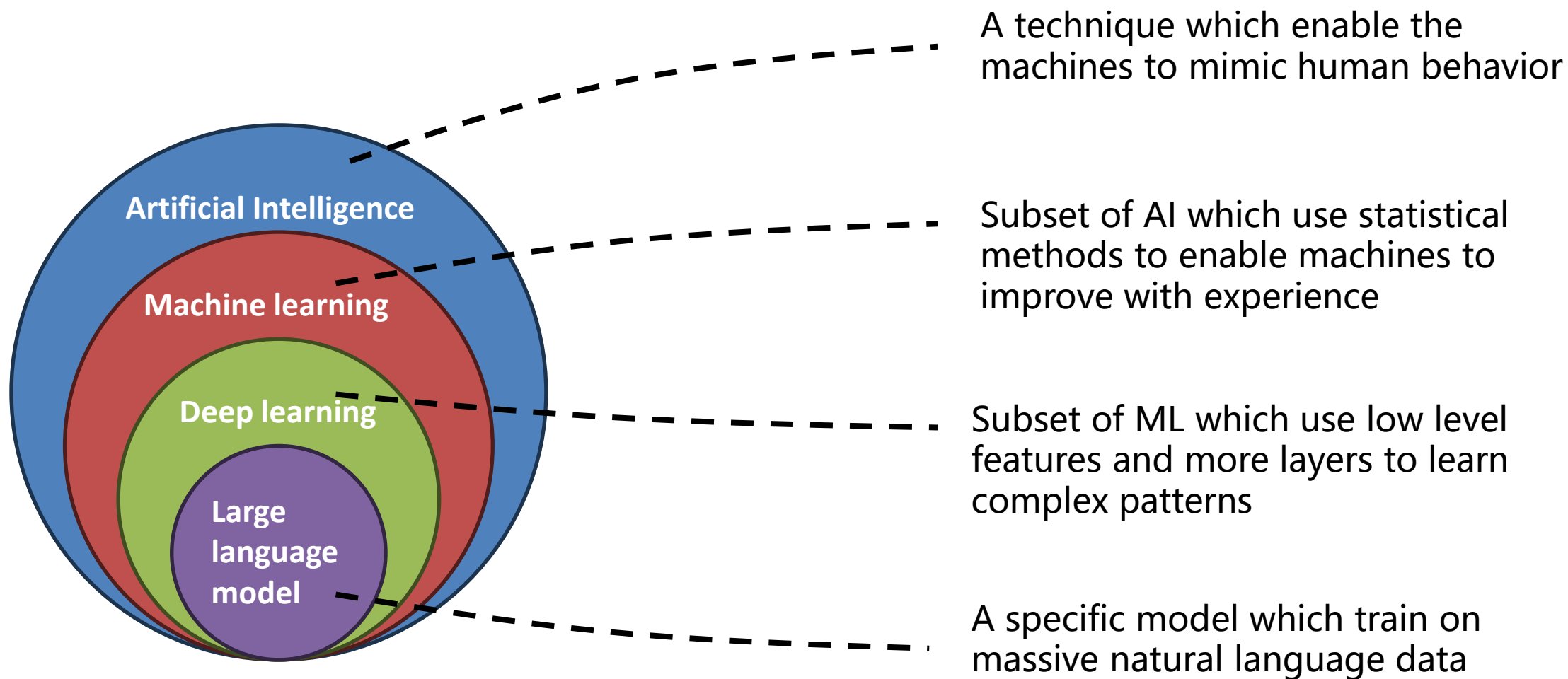
DOI: [10.1016/0168-9002\(92\)90011-R](#)

Report number: CERN-PPE-92-098, CERN

View in: [CERN Document Server](#)

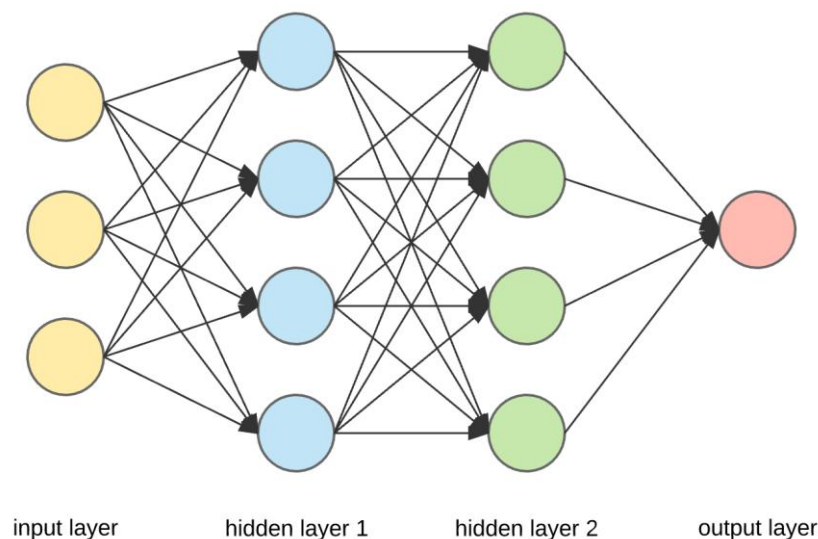
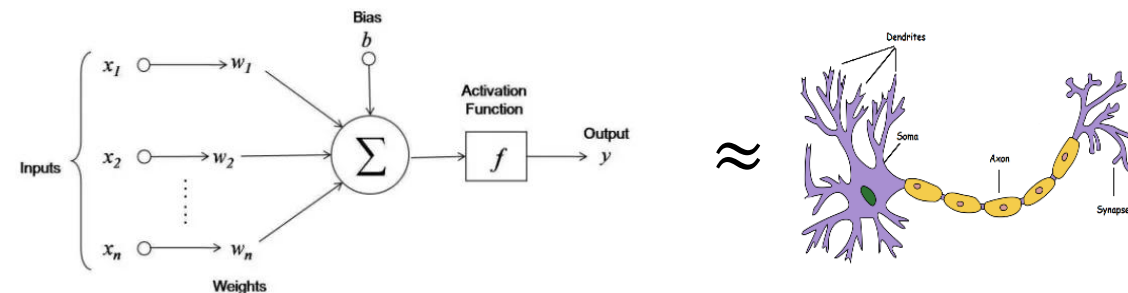


AI, ML, Deep learning

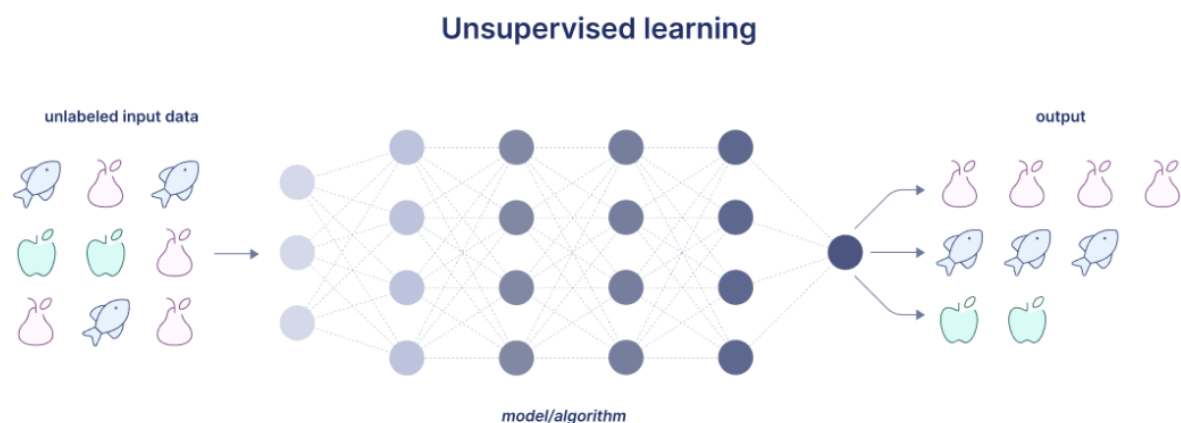
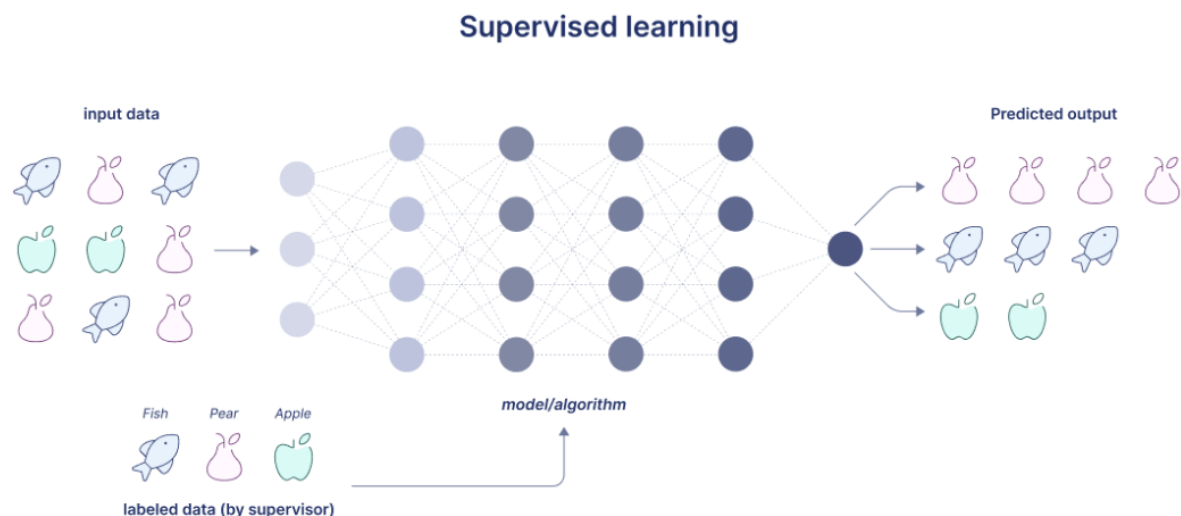


What is Machine learning

- In CS, neuron is a simple function $\sigma(\sum x_i w_i + b)$
- Neural network: multiple connected neurons
 - An extremely complex function which maps the input to output
 - Millions/Billions float parameters
 - Training: adjust the parameters to make outputs move towards target, e.g. truth, similar to fitting
 - Two key parts:
 - Data representation
 - Function (model)



What is machine learning



- **Supervised learning**

- Training a model on labeled data, allowing the model to learn the relationship between inputs and outputs
- Classification, Regression, Gradient Boosting etc..

- **Unsupervised learning**

- Training unlabeled data, identifying patterns and structures within the data without predefined labels
 - Nearest neighbor mapping, k-means clustering, anomaly detection etc..
- Others, semi-supervised learning, reinforcement learning, not covered today

What is machine learning

Traditional approach

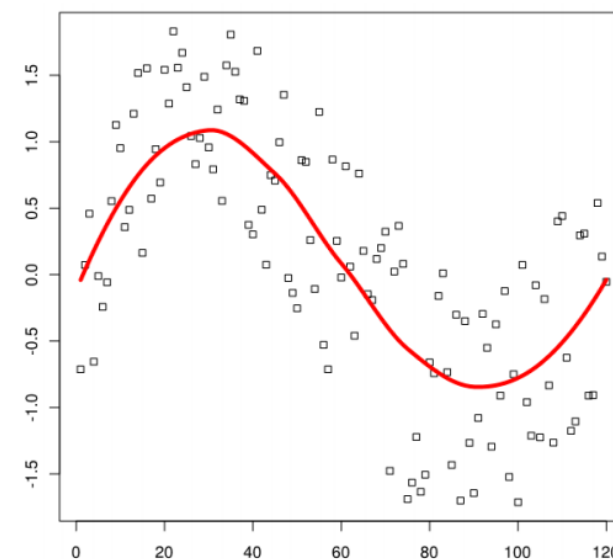


Machine learning



How machine learn

- For traditional fitting:
 - User define a function $y = f(x)$
 - Define a χ^2 or likelihood $\mathcal{L}$
 - E.g. $\chi^2 = \sum (y_i - f(x_i))^2 / \sigma_i^2$
 - **Minimize χ^2 or $-\log(\mathcal{L})$**
- For machine learning:
 - User **define a model** (combinations of multiple neurons) M
 - Define a loss, e.g. $\text{MSE} = \sum (y_i - M)^2$
 - **Minimize loss**

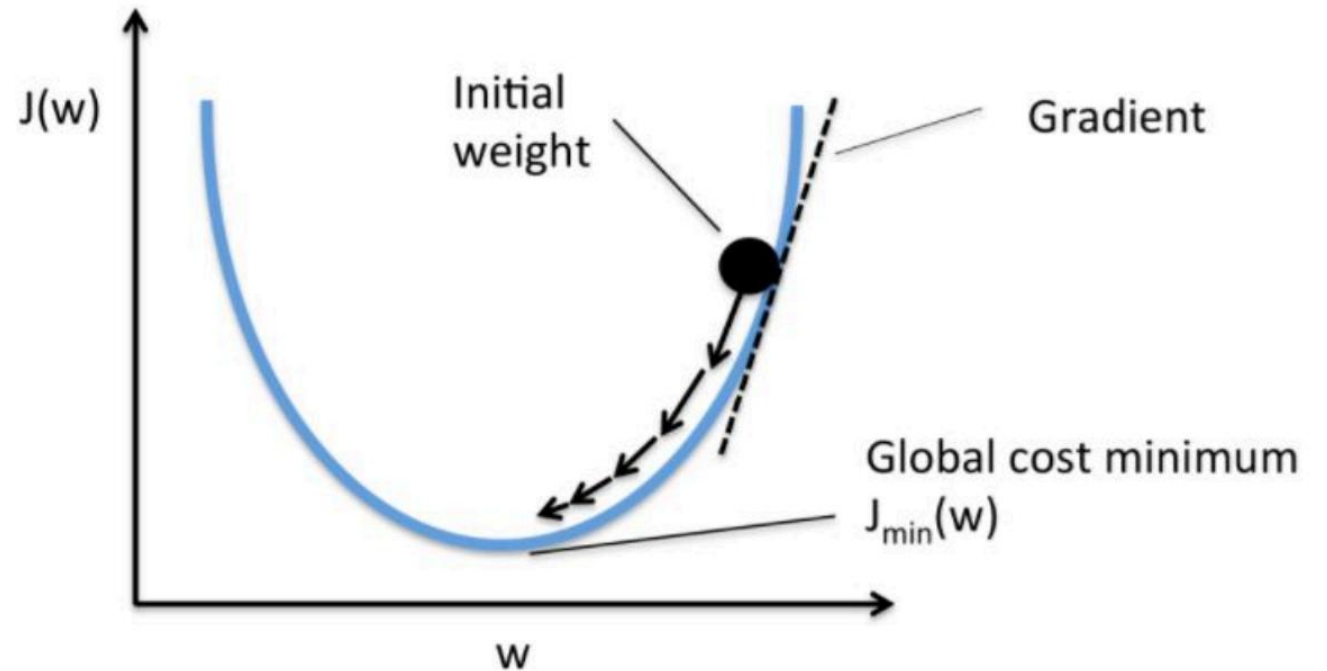


Gradient descent

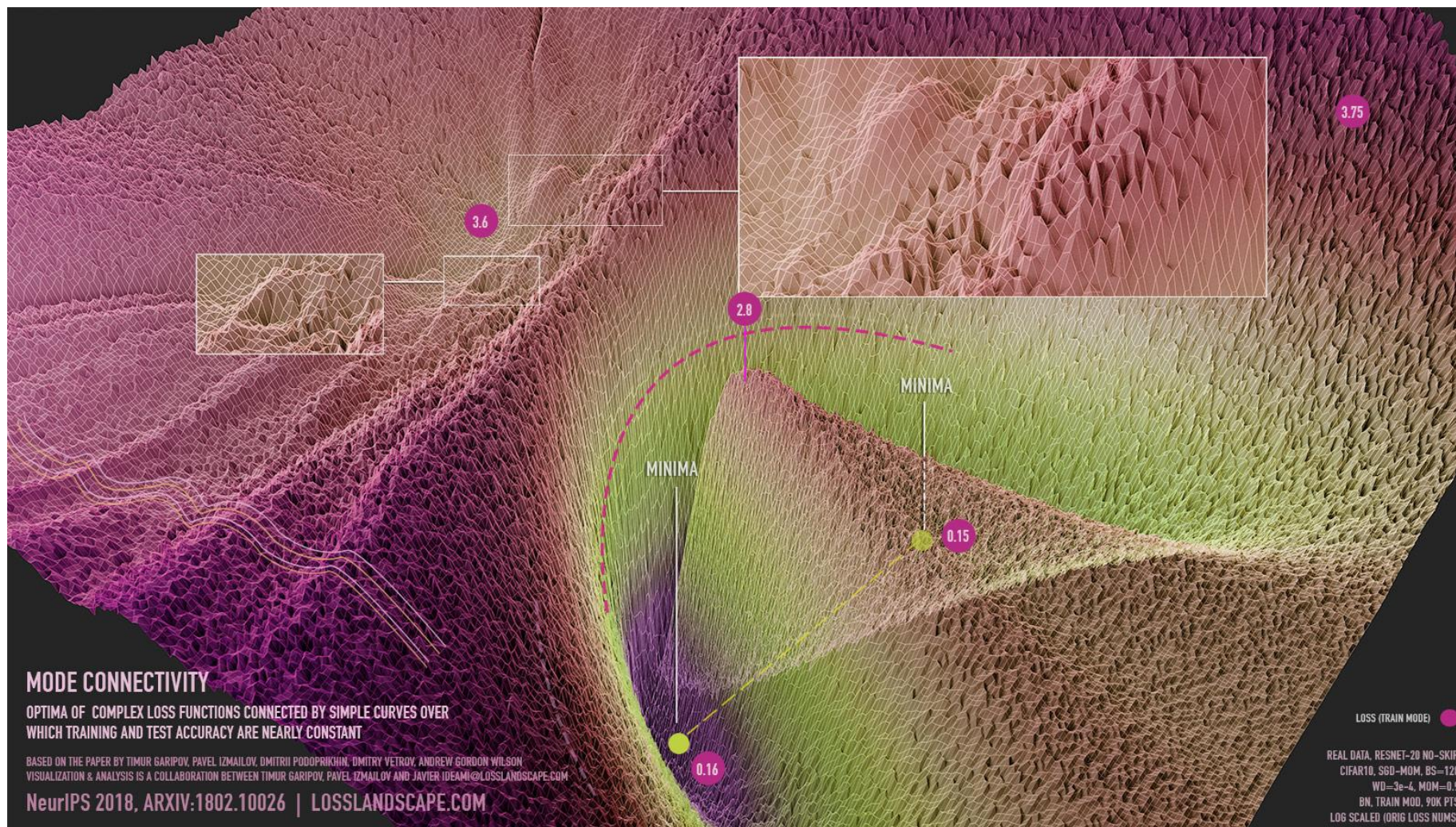
Starting from an arbitrary position, the weights are updated after every iteration "t" ,

$$w_j(t+1) = w_j(t) - \eta \frac{\partial L}{\partial w_j}(t)$$

η is the learning rate

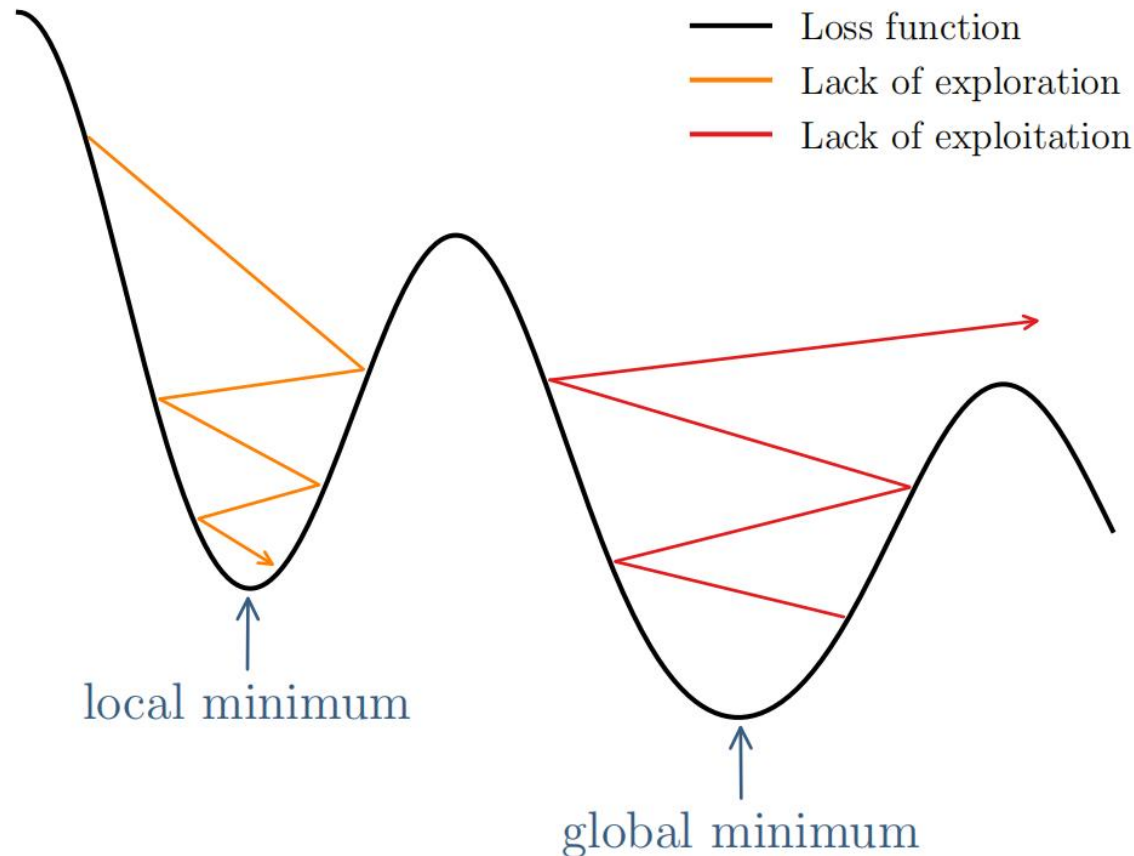


What if we have too many parameters?



Local and global minima

- Always a problem for ML
- Typical solution:
 - Random Initialization and Multiple Restarts
 - Simulated Annealing
 - Noise Injection
 - Dynamic Learning Rates
 - Etc.
- Recommendation
 - Start with Adam + learning rate scheduling + multiple random initializations

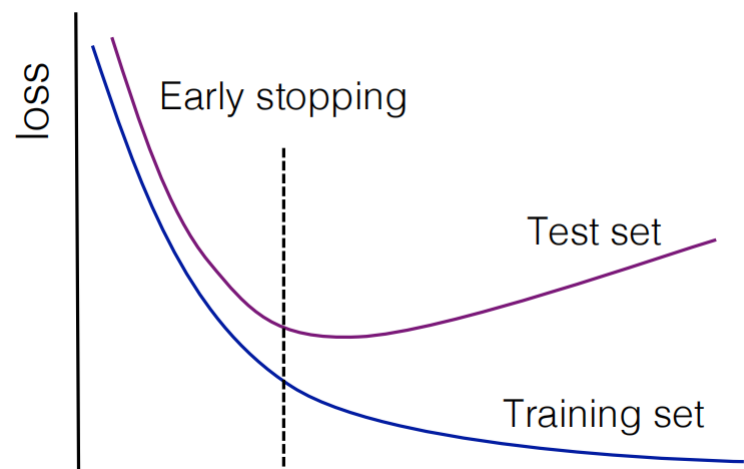
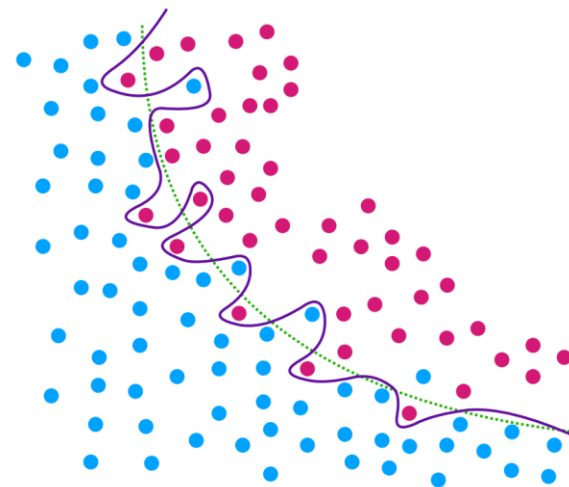


Overfitting and regularization

- More data
- Simpler models
- Dropout
- Early stopping
- Regularization
 - Add penalty terms to the loss function to limit large weights

$$\text{New_Loss} = \text{loss} + \lambda \sum w_j$$

- Recommendation:
 - use cross-validation to select the best regularization method and hyperparameters



Popular ML models

- Boosted decision tree (BDT)
- Multi-layer perceptron (MLP)
- Convolutional NN (CNN)
- Graph NN (GNN)
- Transformer
- Not covered today
 - Recurrent NN (RNN)
 - Generative adversarial network (GAN)
 - AutoEncoder (AE)
 - Flow
 - Diffusion
 - Physics informed NN (PINN)

Useful libraries

- **scikit-learn:** For most of the “basic” algorithms like Linear Regression and Boosted Decision Trees. Also useful for preprocessing, model combination, etc.



- **XGBoost, LightGBM, CatBoost:** For optimized tree-based models

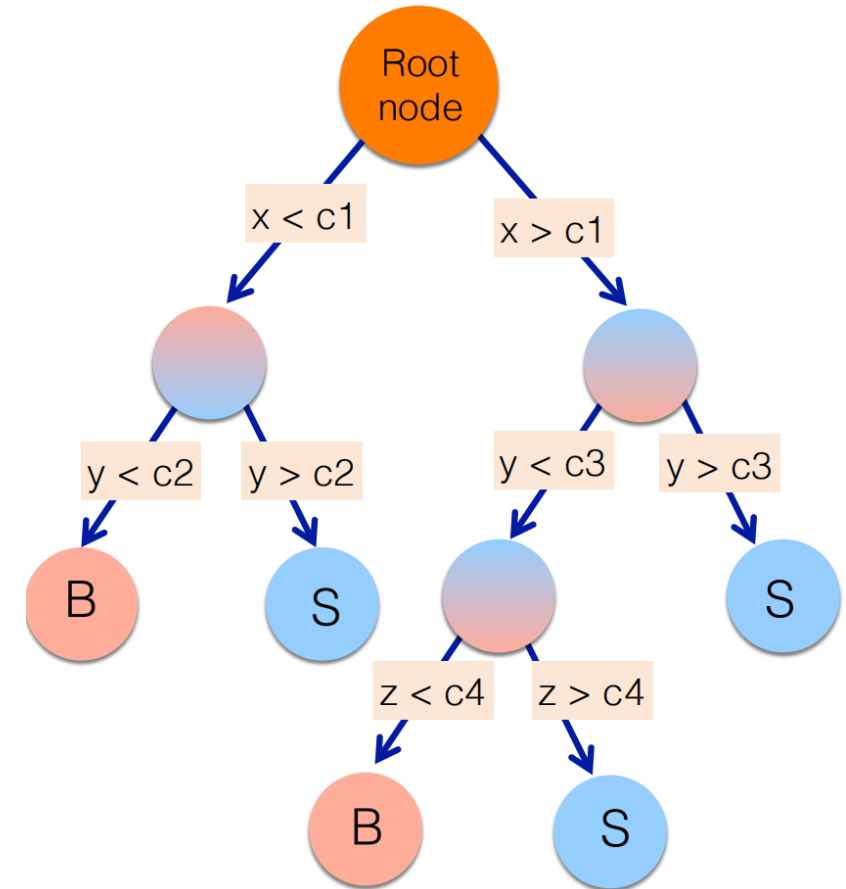


- **TensorFlow, PyTorch, Jax:** For deep learning models

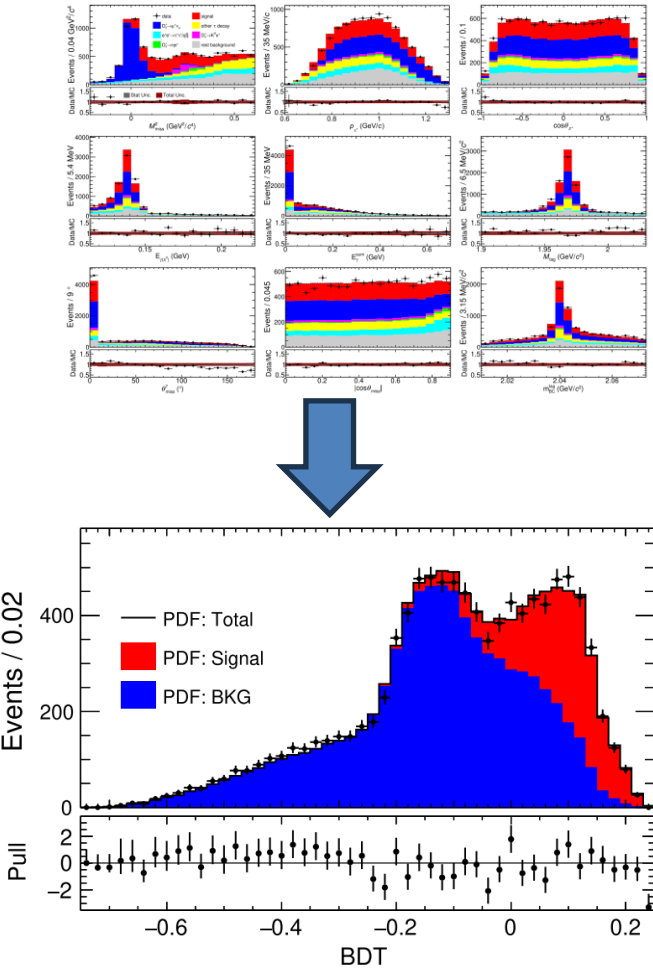


Decision tree

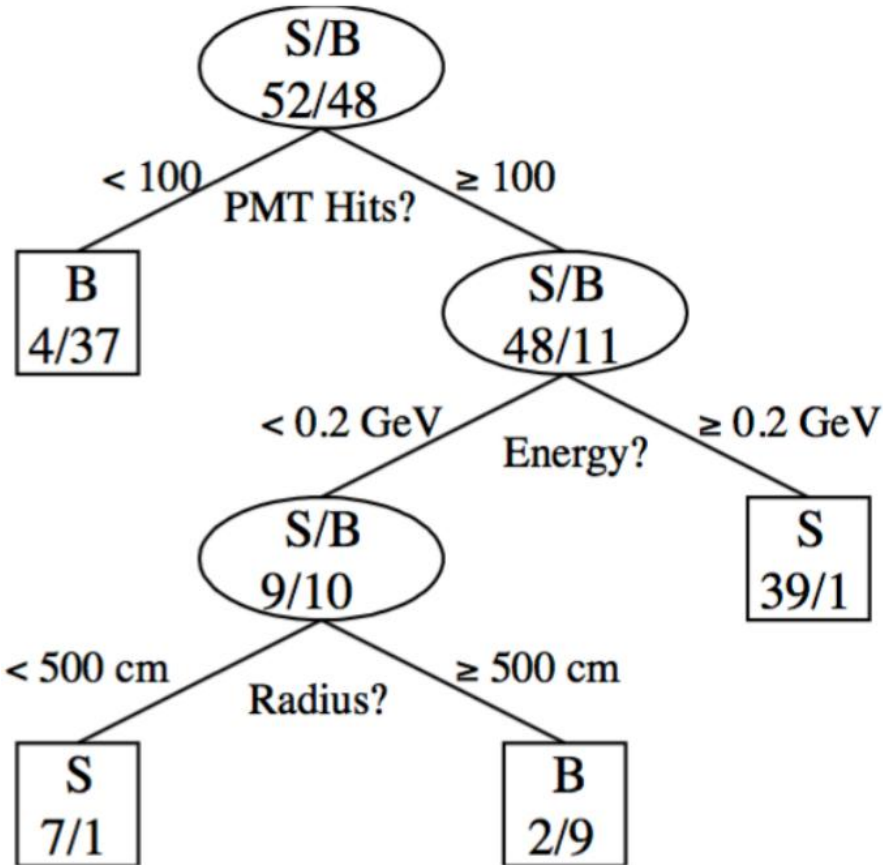
- Sequential application of cuts -> splits the data into nodes
 - Each cut depends on cuts in previous nodes
 - The process is repeated until some condition (depth of tree) is met
- Final nodes classify an event as signal or background
- Boosting: use multiple trees
- Advantage:
 - Easy to understand/interpret
 - Training is fast
 - Good performance with multivariate data



Applications at HEP



Signal/Background



arXiv: [0408124](https://arxiv.org/abs/0408124)

Multilayer perceptron (MLP)

- Classic neuron network
 - 1980s
- Fully connected network
- Still popular, e.g. feedforward block in transformer

Many options for activate function

Function	Formula	Use Case
ReLU	$f(x) = \max(0, x)$	Hidden layers (avoids vanishing gradients).
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	Binary classification output.
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Hidden layers (zero-centered output).
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	Multiclass classification output.

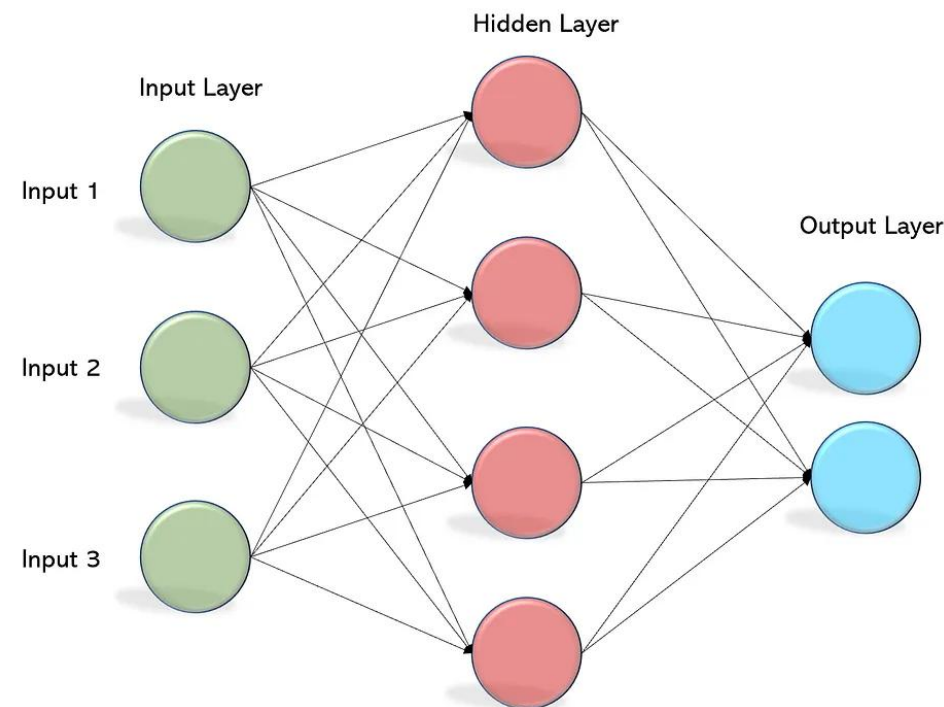
And loss

- **Classification:** Cross-Entropy Loss

$$L = - \sum_{i=1}^C y_i \log(p_i)$$

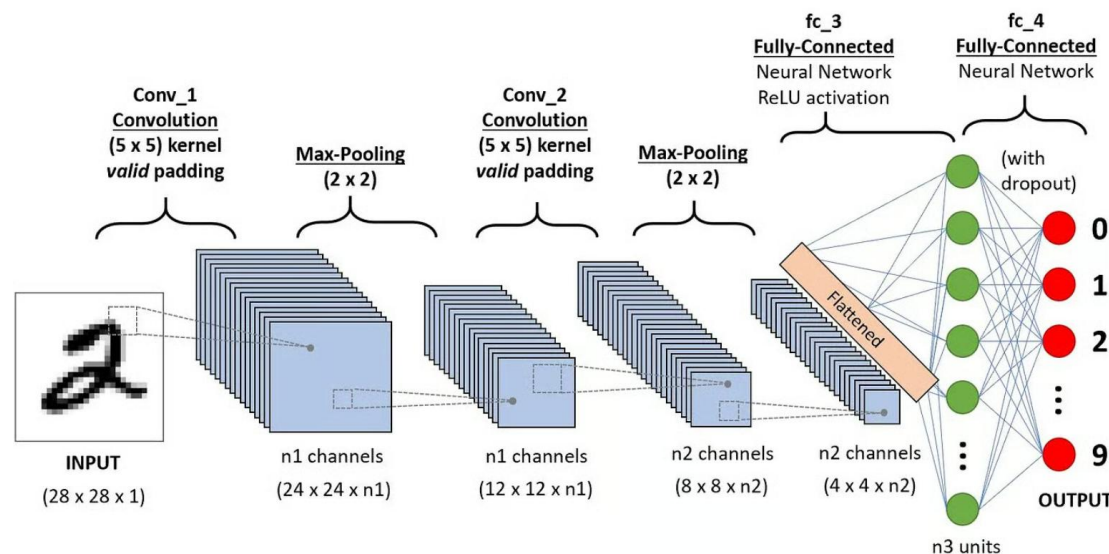
- **Regression:** Mean Squared Error (MSE)

$$L = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



Convolutional NN (CNN)

- Inspired by the layered architecture of the human visual cortex
- Use **convolution kernel to identify and extract patterns and features**
 - All kernels move over the whole image
 - One kernel is used to recognize one pattern**, e.g. one kernel to recognize a straight line, another for vertical line
- Use max/average pooling to do dimensionality reduction while preserving salient features



Convolution:

$$y[n] = (f * g)[n] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[n - k]$$

2D Convolution:

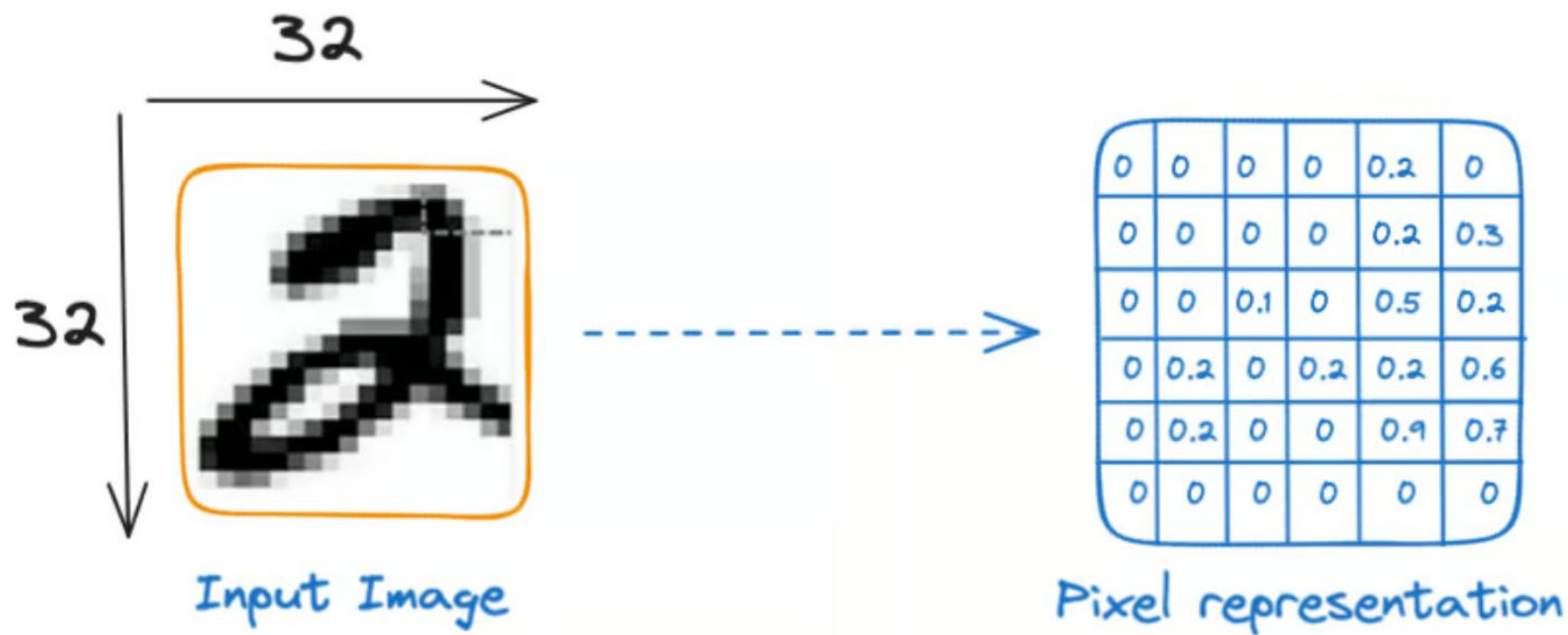
$$y[i, j] = (f * g)[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] \cdot g[i - m, j - n]$$

CNN convolution kernel:

$$(f \star g)[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] \cdot g[m - i, n - j]$$

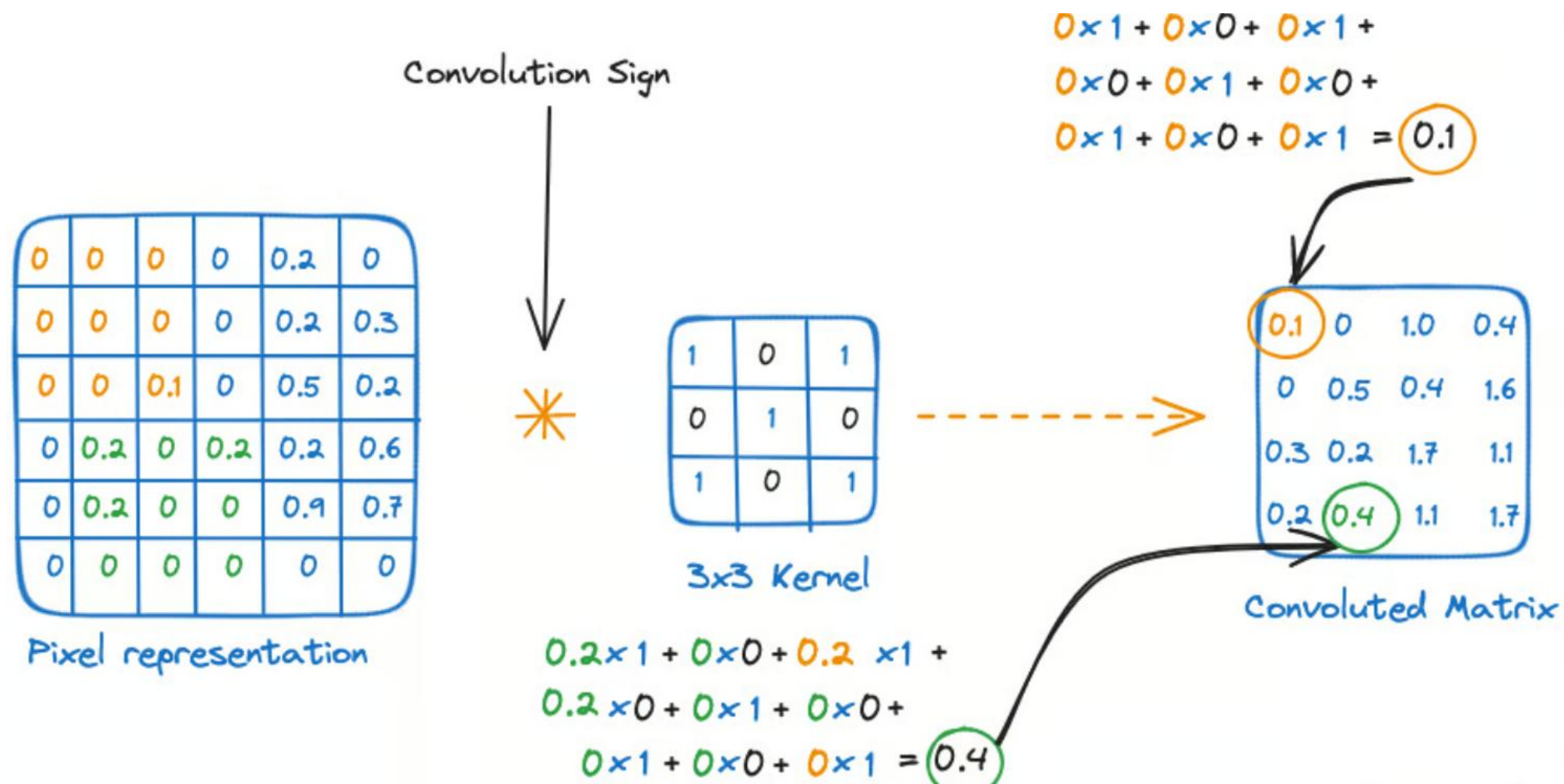
19

CNN, take a closer look



More details [in tutorials](#)

CNN, how convolution works



CNN, pooling



Feature extracted

CNN summary

Prons	Cons
Spatial Hierarchy Preservation	Fixed Kernel Size
Same kernel slides across input, reducing parameters.	Struggles with Non-Grid Data
Detects features regardless of position	Pooling Loss
Efficient for Grid Data	Fixed Input Size

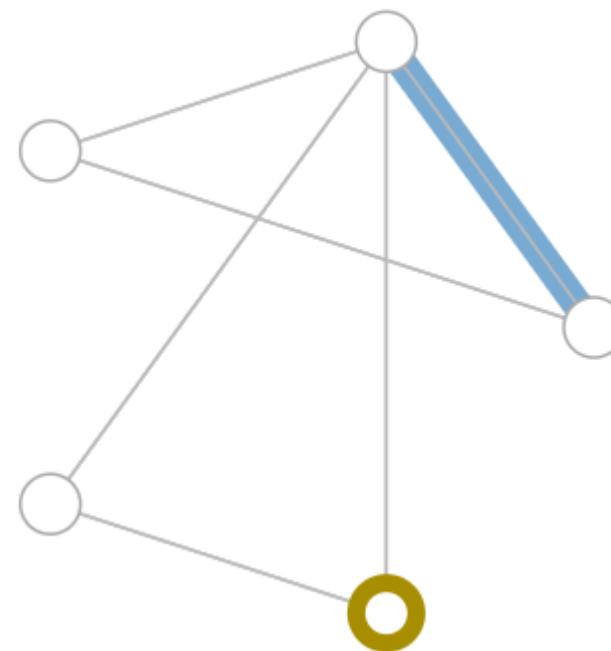
Graph NN (GNN)

GNN vs CNN

Property	GNN	CNN
Data Topology	Non-Euclidean (graphs)	Euclidean (grids)
Core Operation	Message passing	Sliding kernel convolution
Parameter Sharing	Shared across nodes	Shared across spatial locations
Translation Invariance	Node-order invariant	Partially invariant to shifts
Typical Tasks	Node classification, link prediction	Image classification, object detection

Key features:

- **Nodes (Vertices):** Represent entities (e.g., users in social networks, atoms in molecules).
- **Edges:** Represent relationships between nodes (e.g., friendships, chemical bonds).
- **Aggregation:** Each node collects features from its neighbors (e.g., mean, sum).
- **Update:** Combines its own features with aggregated neighbor features.
- **Iteration:** Repeats to capture higher-order neighborhood information.



GNN, take image as an example

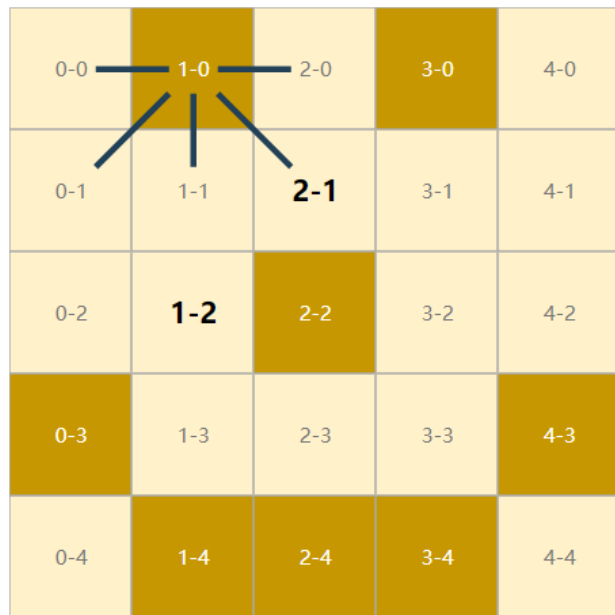
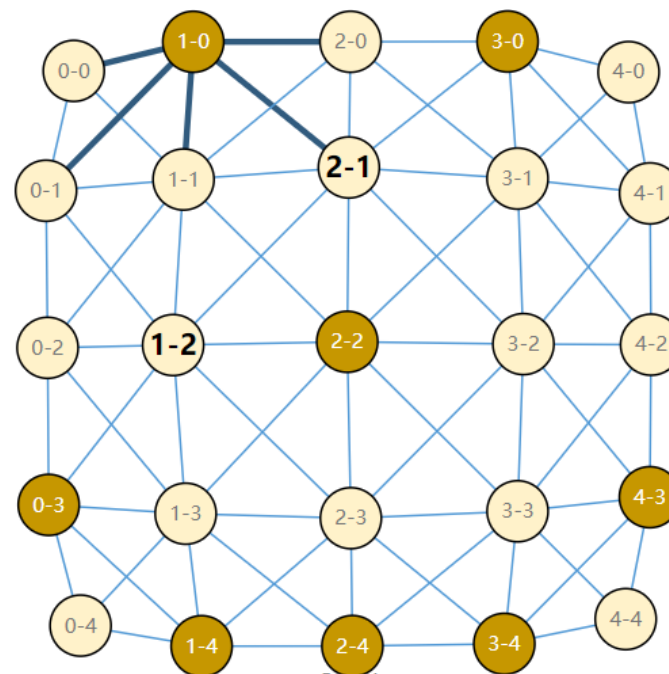
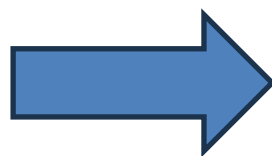


Image Pixels

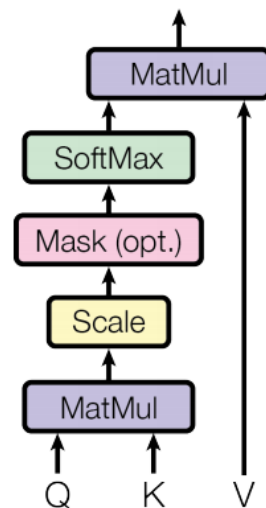


In GNNs, **all nodes can be interconnected**, unlike in CNNs where only spatially adjacent nodes are connected.

Transformer and attention

- **Query (Q)**
 - Represents "what I'm looking for. "
- **Key (K)**
 - Represents "what I have."
- **Value (K)**
 - Represents "the information I want to get if a match is found"

Scaled Dot-Product Attention



Example: You're at a Large Party Looking for Interesting People to Chat With

- **Q:** You walk into the crowd with your interests and topics you want to discuss.
- **K:** Every person at the party is implicitly carrying their self-introduction and tags
- **V:** When your "query" matches with a person's "key," the specific content (their experiences, knowledge, insights) you want to gain is the "value."

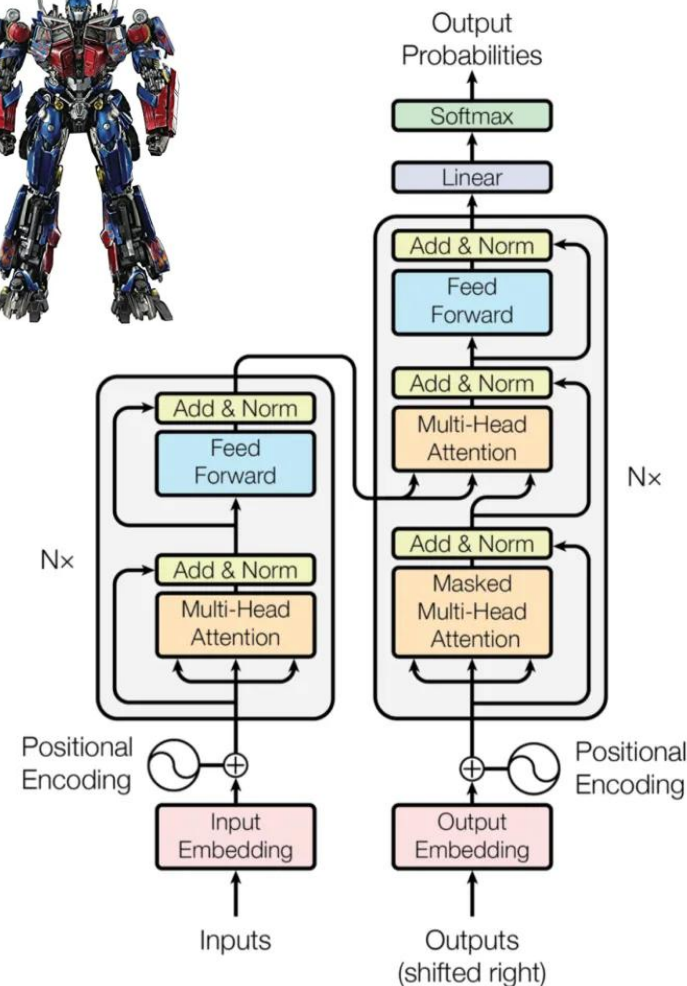


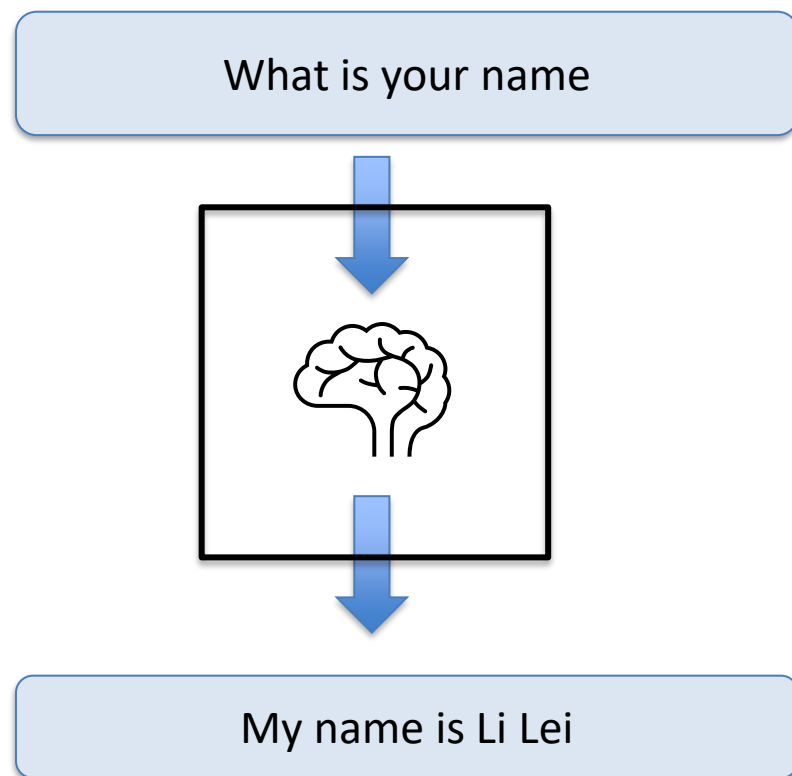
Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention Is All You Need." *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

Citation> 26万 !

LLMs & Agents

Large Language Model (LLM)

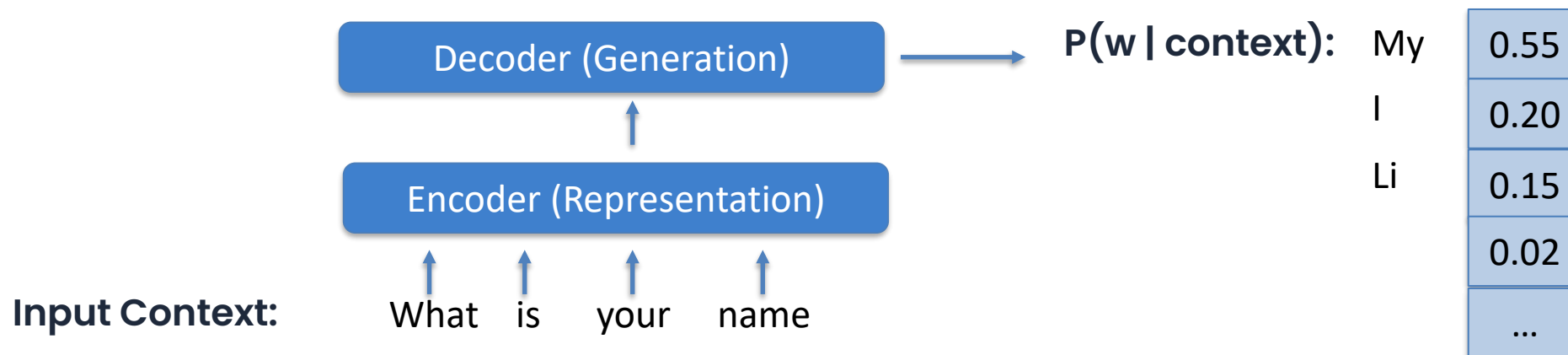
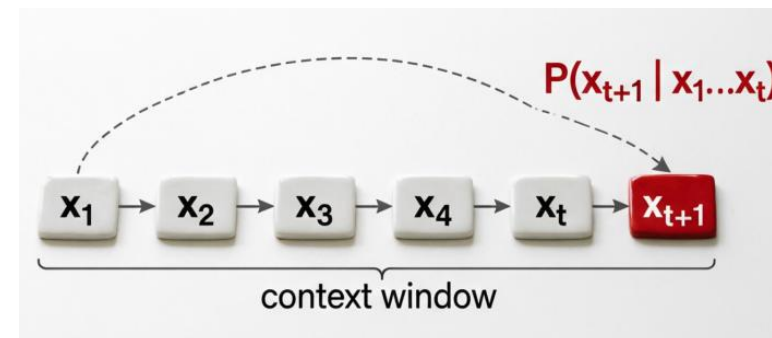


What is LLM

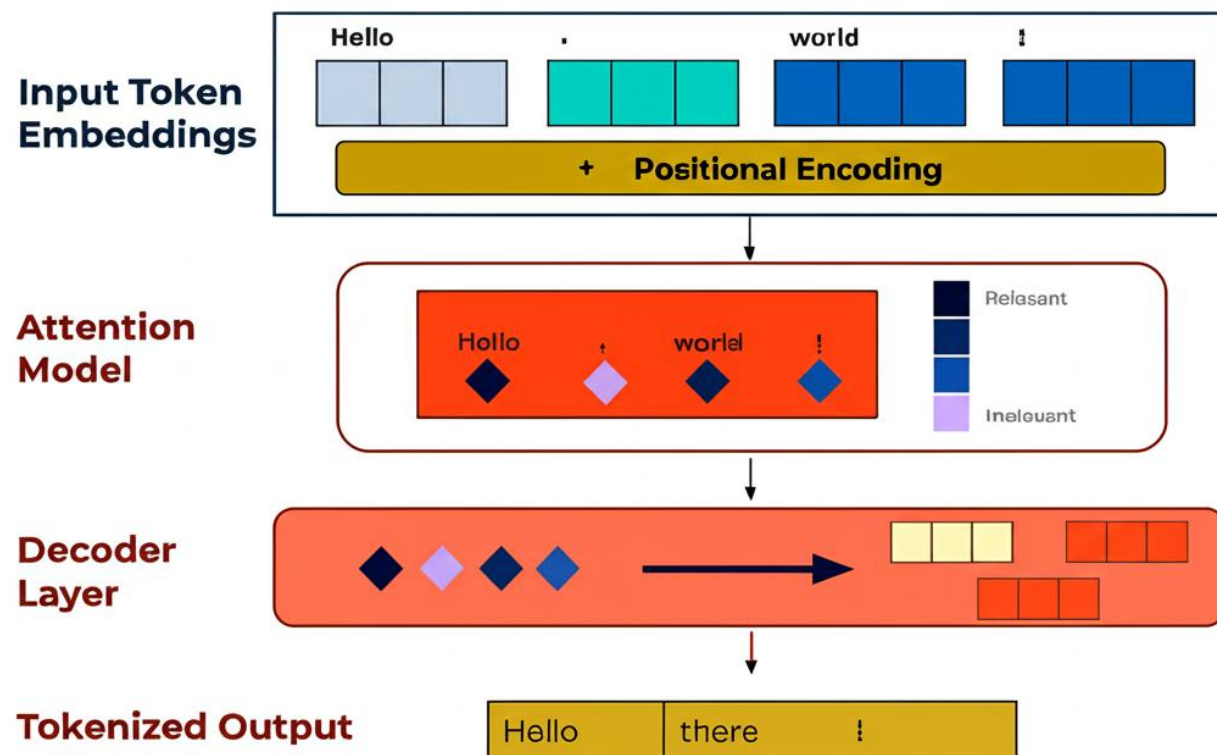
A very large Neural Network:

Input: a sequence of words (tokens)

Output: a distribution over possible next words



What is LLM



- **Trillions of parameters** in the model
- Trained on massive amount of text data
- Predict the next most **statistically** probable token
- This training allow the model **“understand” the patterns** in the training data

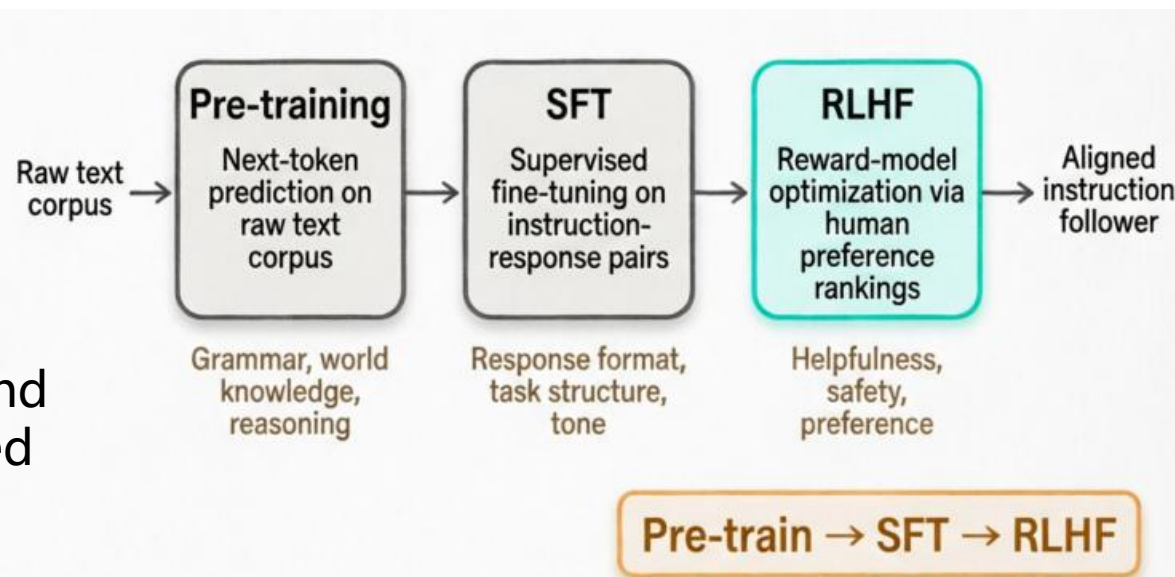
We should embed our knowledge into these patterns

Why LLM matters for Science

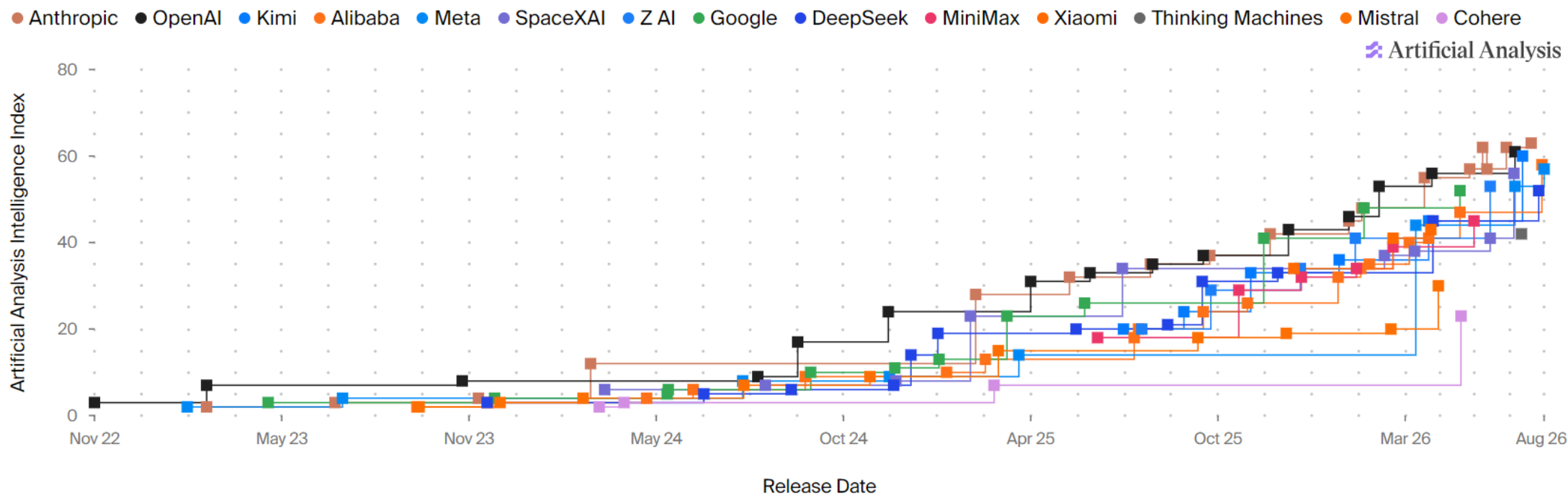
- For centuries, understanding and generating human language has been a uniquely human trait.
- It's how we express ideas, build relationships, and create knowledge.
- For computers, however, it's been an incredibly complex challenge due to language's nuance, ambiguity, and context.
- We always “translate” our knowledge, our understanding into natural language !
- If we want to build an “intelligent” system, we have to use LLM otherwise it' s difficult to communicate with human !

Training pipeline

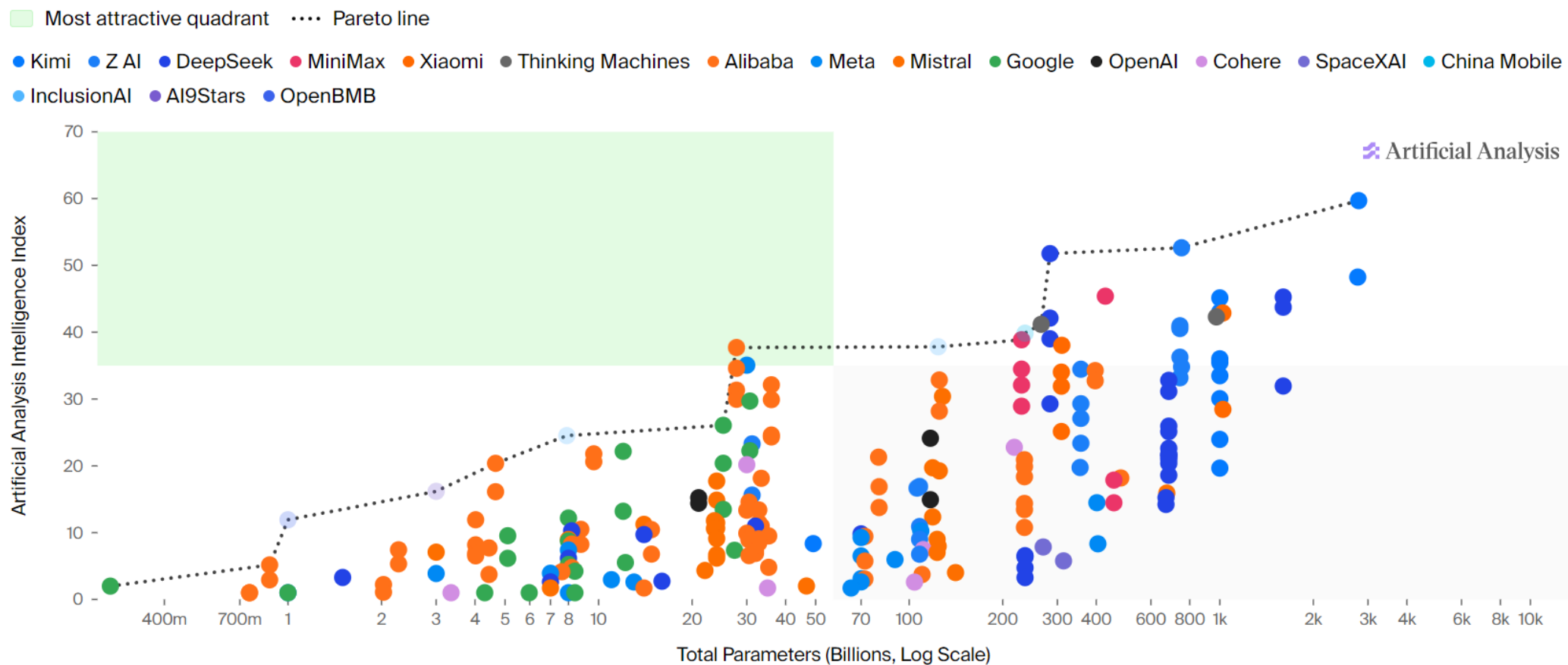
- Normally LLMs are trained on an **absolutely massive amount of text data** – billions, even trillions, of words from:
 - Books
 - Articles
 - Websites
 - Conversations
 - And more!
- This training allows them to “understand” and generate human language with unprecedented accuracy.



AI trends



AI trends



More details in <https://artificialanalysis.ai/trends>

How can LLM help us to do research

- What Can LLMs Do?
 - Answer questions: From simple facts to complex explanations.
 - Summarize long documents: Condense articles, reports, or research papers.
 - Generate code: Assist programmers by writing or debugging code snippets.
- **It's Not Magic, It's Statistics!**
 - Important to remember: LLMs aren't truly "thinking" or "understanding" in the human sense.
 - They are incredibly sophisticated statistical models.
 - When you ask an LLM a question, it's not consulting a knowledge base like a human brain.
 - Instead, it's predicting the most statistically probable sequence of words that would answer your question based on all the data it's seen during training.
- Another application: **Automation -> Autonomy discovery**
 - Take LLM as a controller, i.e. generate configuration/programs to control the scientific tool
 - **DrSai project at BESIII**

What is your name

My name is Li Lei

I am Li Lei

Li Lei

My family name is Li, given name is
Lei

Which one is correct?

Prompt engineering

- Important feature for ML
 - **Garbage in, garbage out !**
 - Same for the prompt of LLMs

Less effective ❌:

```
Extract the entities mentioned in the text below. Extract the following 4 entity types: company names, people names, specific topics and themes.
```

```
Text: {text}
```

Show, and tell - the models respond better when shown specific format requirements. This also makes it easier to programmatically parse out multiple outputs reliably.

Better ✅:

```
Extract the important entities mentioned in the text below. First extract all company names, then extract all people names, then extract specific topics which fit the content and finally extract general overarching themes
```

```
Desired format:
```

```
Company names: <comma_separated_list_of_company_names>
```

```
People names: -||-
```

```
Specific topics: -||-
```

```
General themes: -||-
```

```
Text: {text}
```

零样本提示

少样本提示

链式思考 (CoT) 提示

自我一致性

生成知识提示

Prompt Chaining

思维树 (ToT)

检索增强生成 (RAG)

自动推理并使用工具 (ART)

自动提示工程师

Active-Prompt

方向性刺激提示

Program-Aided Language Models

ReAct框架

Reflexion

多模态思维链提示方法

基于图的提示

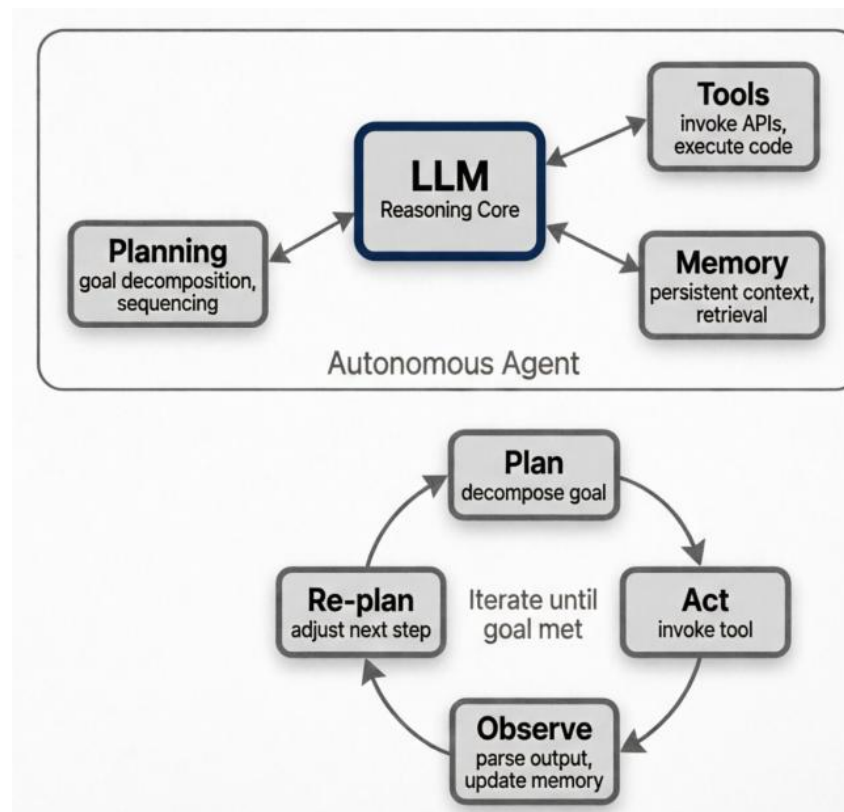
AI Agents

A LLM + a well controlled environment

Input: a task (a sequence of words)

Output: results

- Plots
- Numbers
- ...



LLMs vs. AI Agents

Large Language Models (LLMs)

Passive Knowledge Engines: Process text inputs and generate predictions based on probabilistic patterns learned during pre-training.

Stateless & Isolated: Cannot execute actions, access live external data natively, or persist memory across disconnected sessions.

Example: Answering a question, drafting an essay, or translating code snippets.

Autonomous AI Agents

Active Task Executors: Combine LLMs as reasoning backbones with tools, memory, and goal-directed planning loops.

Stateful & Action-Oriented: Can browse the live web, execute code, call APIs, continuously self-correct, and achieve complex goals.

Example: Conducting market research, debugging repos, or running multi-day experiments.

Core Architecture of an Agent



Brain / LLM

Acts as the central cognitive engine responsible for reasoning, intent understanding, decomposition, and decision-making.



Planning

Deconstructs complex goals into sub-tasks (Chain-of-Thought, Tree-of-Thought) and self-reflects on execution mistakes.



Memory

Stores short-term working context (chat history) and long-term knowledge (vector DBs, external file systems).



Tools / Action

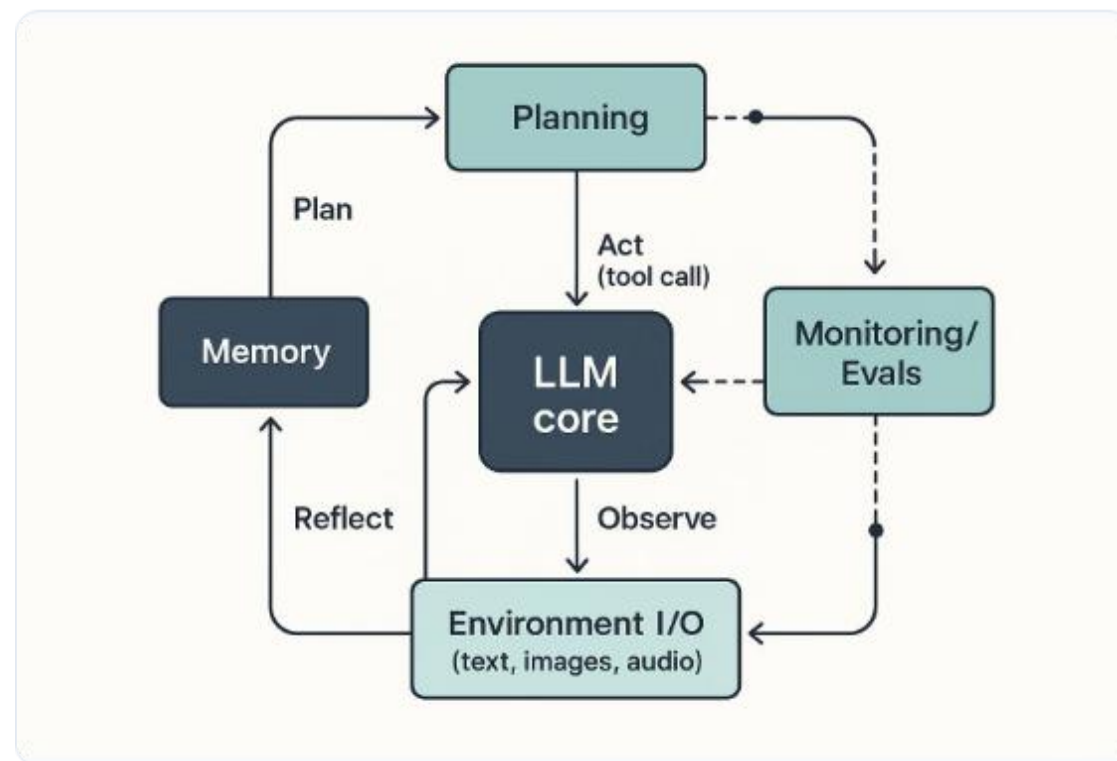
Interfaces with external environments: search engines, Python REPLs, database connectors, and web browsers.

The Agentic Execution Loop

Think → Execute → Reflect

Unlike standard single-pass LLM prompts, AI agents operate in an iterative feedback loop:

1. **Perception & Goal Setup:** The agent receives an goal and analyzes environmental state.
2. **Tool Invocation:** The LLM selects a tool (e.g., web search, code executor) and generates parameters.
3. **Observation & Reflection:** Reads tool output, checks for errors, adjusts its strategy, and iterates until completion.



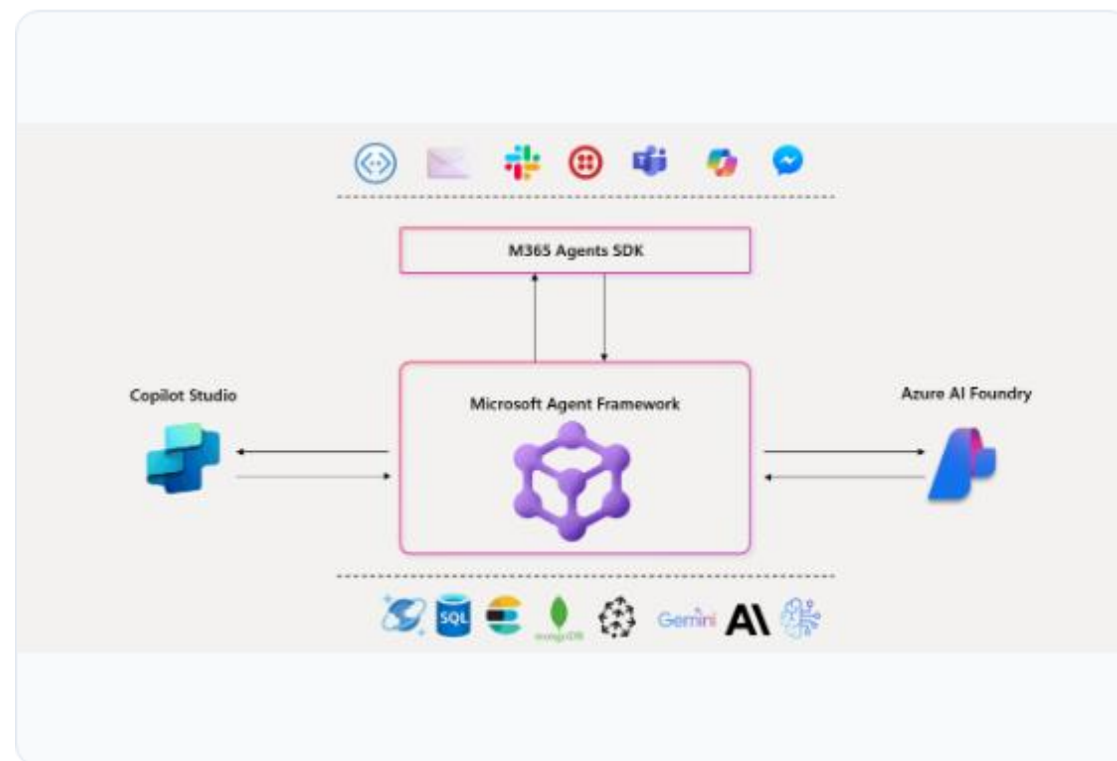
OpenClaw: Autonomous Gateway

Good for personal assistant

Seamless Messaging Integration: Connects AI models directly to everyday channels like Wechat.

Persistent Identity & Memory: Treats agents as long-term digital assistants with dedicated state, skills, and behavior rather than single session chatbots.

Open Skill Marketplace (ClawHub): Extends agent capabilities with one-click installable skills for web research, code execution, and daily tasks.



Deep Research: Multi-Agent Synthesis

Supervisor & Sub-Agent Orchestration

Task Scoping & Delegation: A supervisor agent decomposes complex queries into independent sub-topics and assigns them to isolated sub-agents.

Parallel Investigation: Sub-agents execute tool calls, search web sources, and prune raw tokens before returning clean findings to the supervisor.

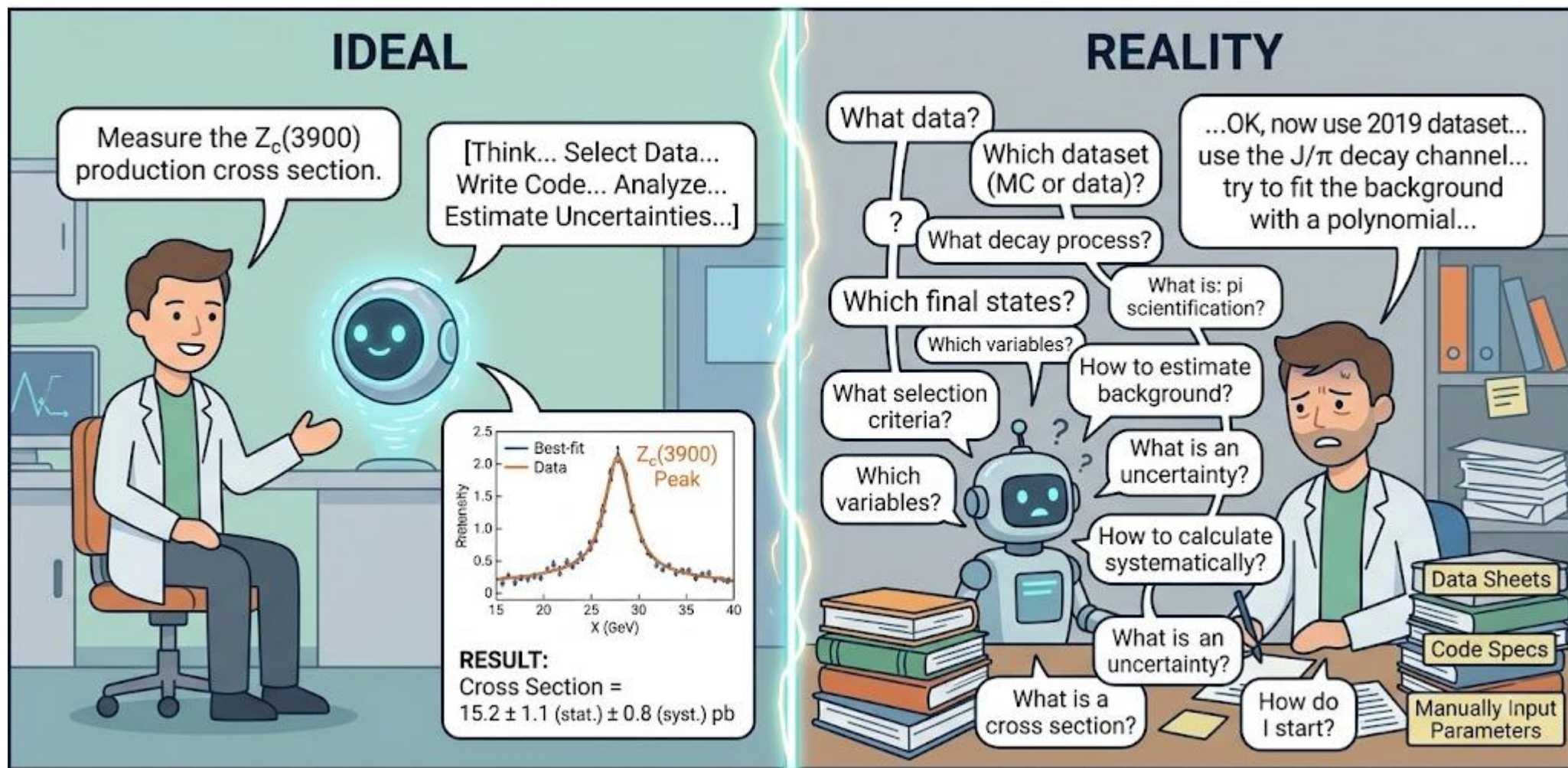
Continuous Reflection & Reporting

Gap Analysis: The supervisor reviews sub-agent results, identifies missing information, and spawns follow-up iterations.

Structured Synthesis: Produces comprehensive reports complete with methodological transparency, confidence ratings, and citations.

Many many Challenges !

AI AGENT IN PARTICLE PHYSICS: IDEAL VS. REALITY

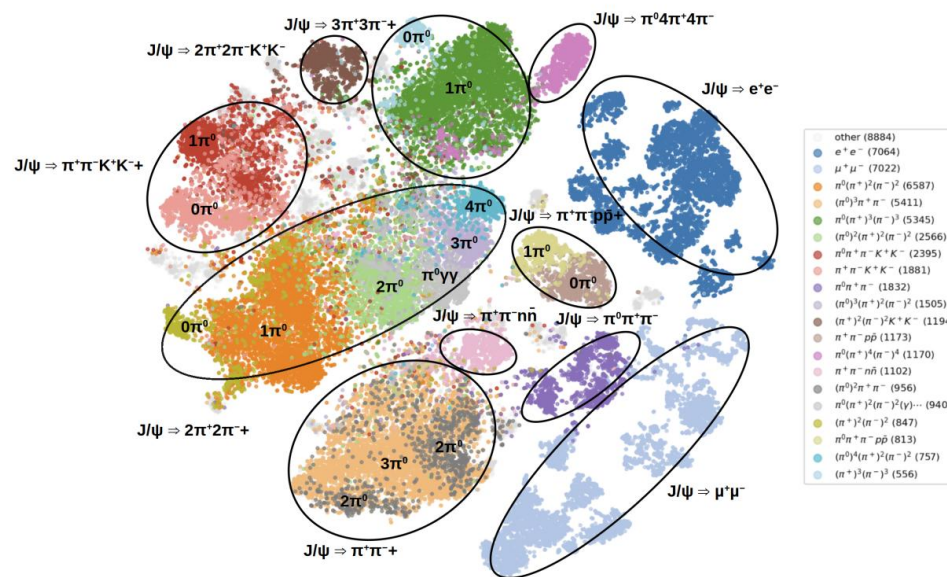
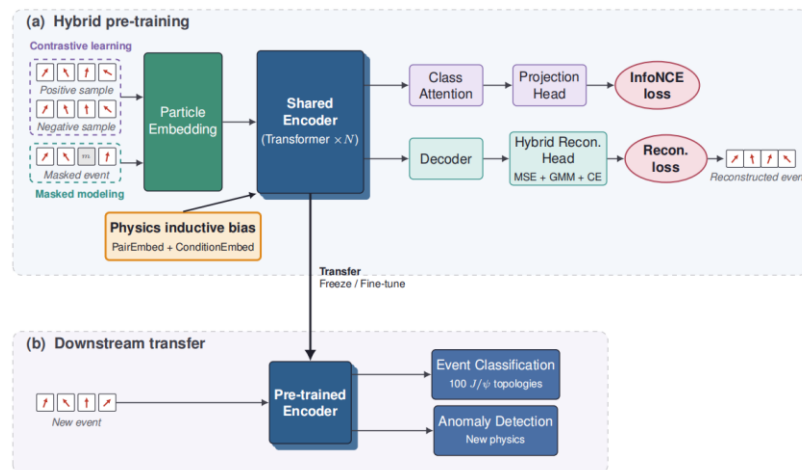
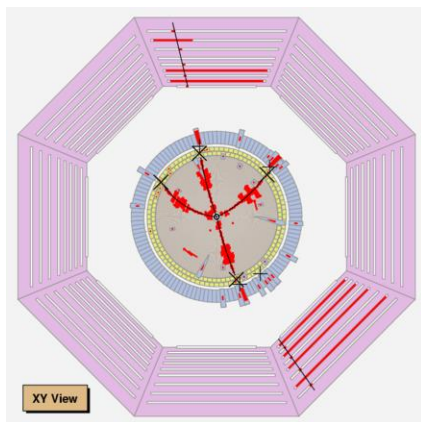


Questions & Discussion

backup

基于实验数据的Foundation Model

- 海量高维数据的**统一表征**
 - 数万个读出通道
 - 超百亿对撞事例
 - 数千/数万个物理过程
- 自监督学习和对比学习
 - 学习探测器重建的多个粒子间关联
 - 利用对比学习约束表征空间
 - “**物理意义**”上对齐

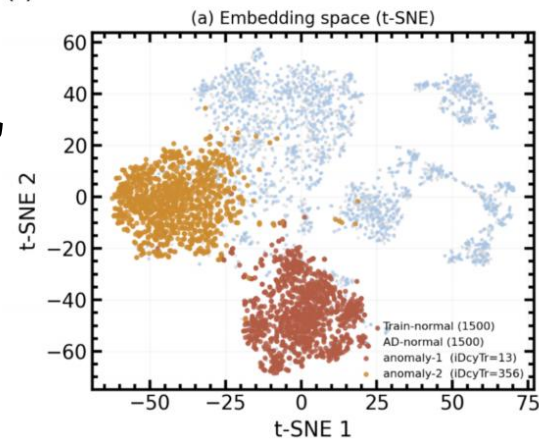


基于实验数据的Foundation Model

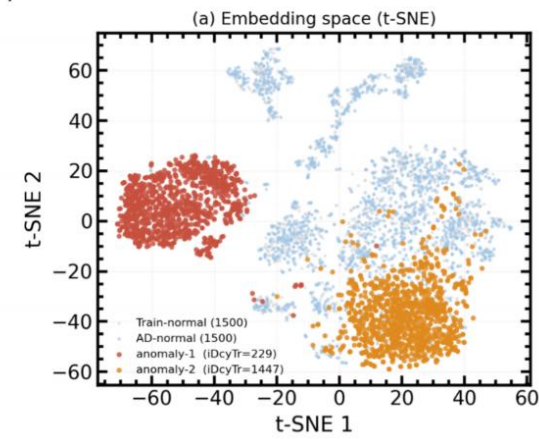
- 相比于传统方法-手动选择本底和训练专用模型，FM的优势：
 - 增加S：信号效率更高，天然涵盖所有过程
 - 降低B：背景排除率更高，背景事例也可分类
 - 支持近乎所有下游分析任务
- 对于训练样本中没有出现的“新粒子”
 - 作为异常信号自动聚类
 - 不同的“新粒子”或“新物理过程”被模型识别并分为两类！
- 目前仅用千万对撞事例训练，计划扩展至亿或十亿

Final state	Method	Precision	Recall	F_1
$\pi^+ \pi^- p \bar{p}$	Cut1	6.3%	56.5%	11.3%
	Cut2	99.5%	50.9%	67.3%
	CLEAR ($\tau = 0.5$)	82.9%	92.5%	87.4%
	CLEAR (matched)	—	—	—
$\pi^+ \pi^- K^+ K^-$	Cut1	31.9%	55.8%	40.6%
	Cut2	99.3%	46.0%	62.9%
	CLEAR ($\tau = 0.5$)	92.8%	78.0%	84.7%
	CLEAR (matched)	99.3%	59.8%	74.6%
$\pi^+ \pi^- \pi^0$	Cut1	79.0%	51.8%	62.6%
	Cut2	97.7%	37.2%	53.9%
	CLEAR ($\tau = 0.5$)	90.3%	90.9%	90.6%
	CLEAR (matched)	97.7%	46.0%	62.6%

(1). Near-OOD Detection

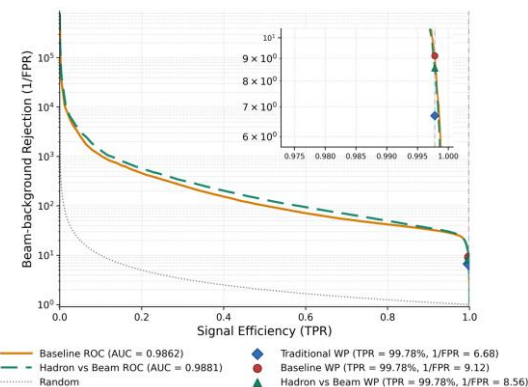


(2). Far-OOD Detection

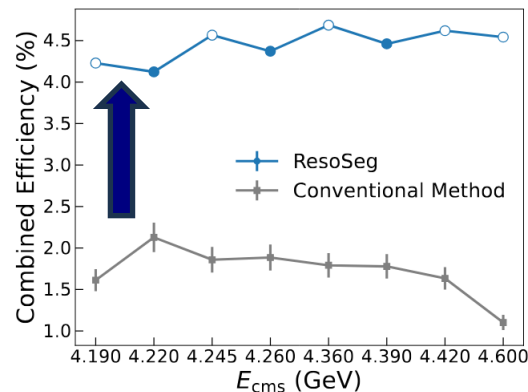
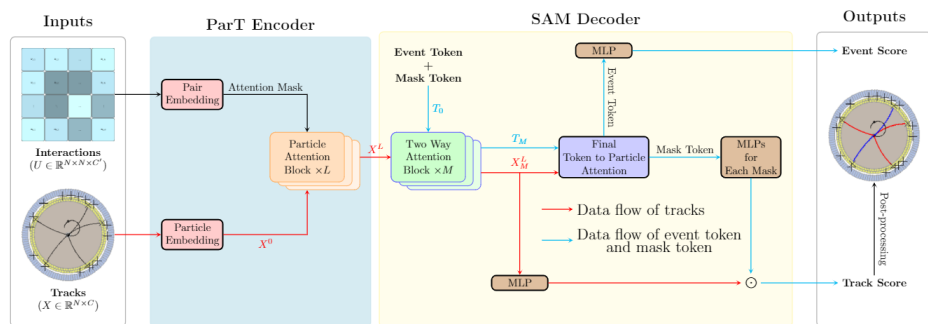
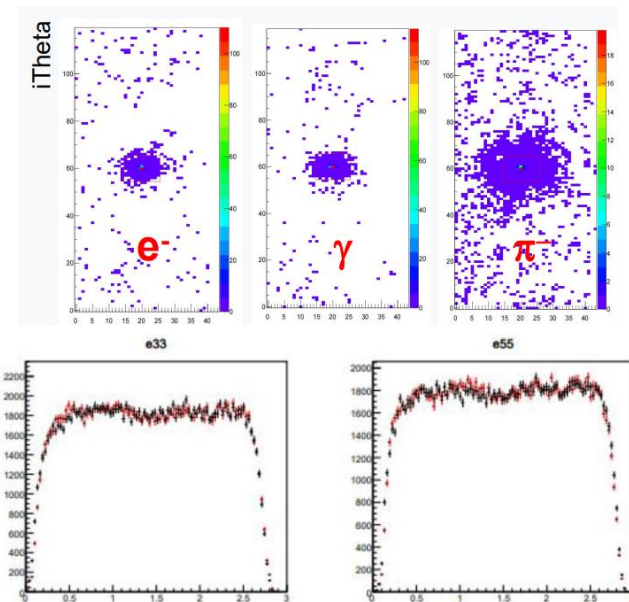


其他代表性深度学习工作

- 在FPGA上部署轻型NN用于快速去除背景
- 基于真实数据训练
- 背景排除提升20-50%**
- 延迟仅35ns
- 对BEPCLII升级和下一代对撞机尤其重要



- 利用MoE+cGAN快速模拟探测器相应
- 与传统Geant4模拟效果相当，速度提升**>100倍**



通用强子标记器

- 利用Transformer训练表征空间
- 利用SAM模型分割末态粒子
- 效率相比于传统方法提升2.6倍**