IEAP CTU in Prague

Data analysis & AI

Yury Shitov, IEAP CTU

7/7/2025

Data analysis of Big Data

The Digital Revolution

- Big Data emerged as a result of the **digital revolution** and the exponential growth in data generated by digital systems.
- **It became a core part of the online economy and transformed how we live, work, and make decisions.**

Two types of Big Data:

- **Canonical (Structured) Data**:
  Planned, systematically collected data (e.g. census, surveys, scientific research).
- **Exhaust (Unstructured) Data**:
  By-product of technological or business processes, often collected unintentionally.

The challenge: Making sense of vast, diverse, and unstructured data sources.

Where Does Big Data Come From?

Non-traditional Big Data sources:

- **Satellite imagery & media files**
  - Used in environmental monitoring, poverty mapping, migration, agriculture, etc.
  - Enabled by advances in computer vision and lower imaging costs.
- **Mobile operator data**
  - Near-complete population coverage
  - Used for migration studies, disaster response, food security, etc.
- **Internet activity**
  - Social media (Twitter, Facebook)
  - Web scraping (e-commerce, news)
  - GIS services (Google Maps, Trends)
  - **Internet of Things (IoT)**: billions of sensors generating real-time data.

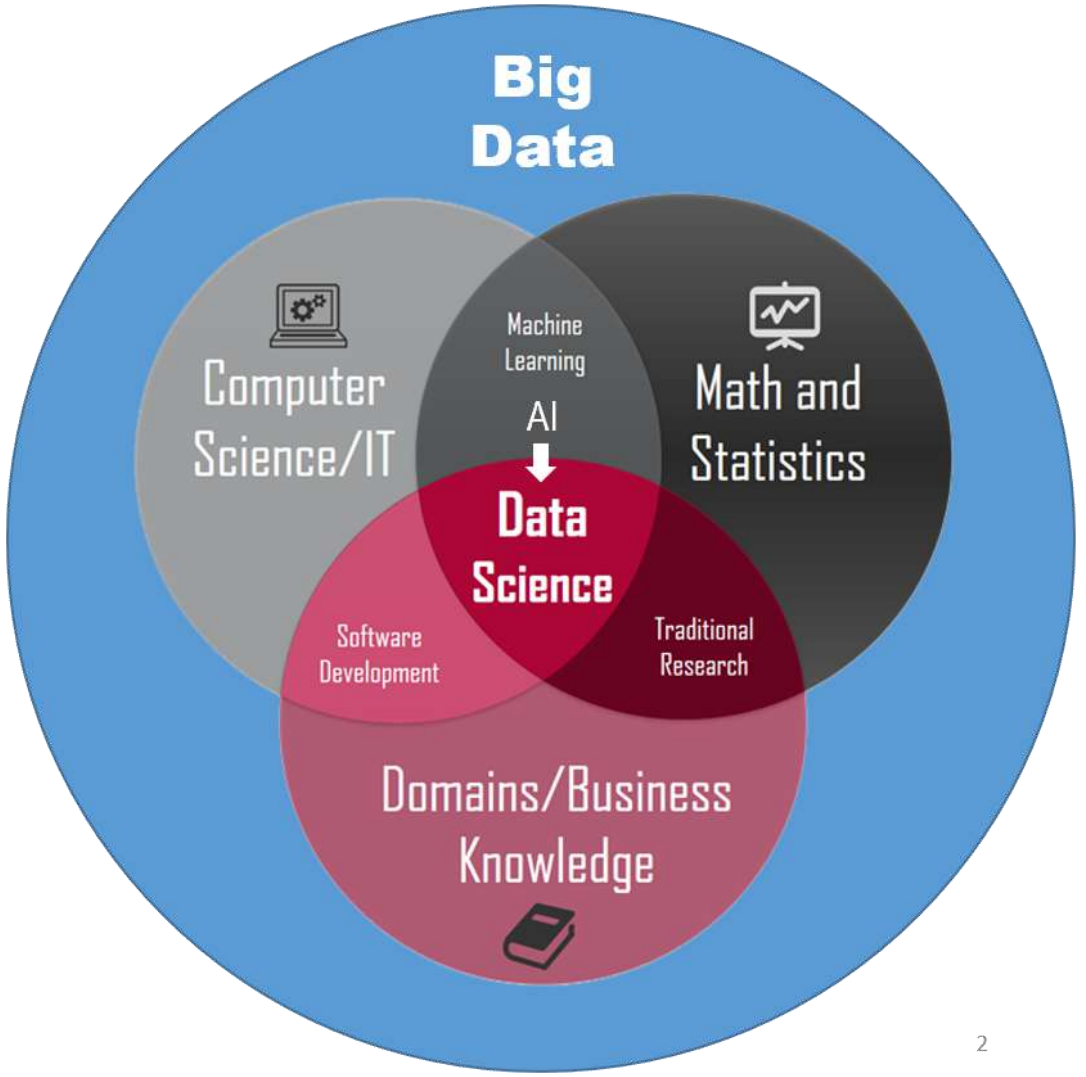
These "digital traces" allow for new insights across disciplines — from economics to public health.

Structure and Terminology

Modern Data Framework

There are **three major foundational areas** driving today's data-driven world:

1. **Computer Science / IT**
   → algorithms, data infrastructure, software tools
2. **Mathematics & Statistics**
   → models, optimization, probability, inference
3. **Domain & Business Knowledge**
   → context, real-world meaning, application relevance



🚦**At Their Intersections:**

Software Development arises from CS and domain knowledge
Traditional Research builds on math and domain understanding
Machine Learning emerged where CS meets math
In the **center**, where all three meet, lies **Data Science**

🤘 **Where Is AI?**

AI originally developed at the intersection of **computer science and math**, focusing on symbolic logic, algorithms, and models
Today, **modern AI (like large language models)** incorporates **domain-specific knowledge** and learns patterns from massive real-world datasets
This shift pulls AI closer to the center — aligning it with the **core of Data Science**

Machine learning (ML)

- **Machine learning** is a field of AI where systems **learn from data** to make predictions or decisions, **without being explicitly programmed** for each task.
- ML methods can be:

1. **Supervised** — learning from labeled data
2. **Unsupervised** — discovering patterns in unlabeled data
3. **Reinforcement** — learning through trial-and-error interactions

Key concepts:

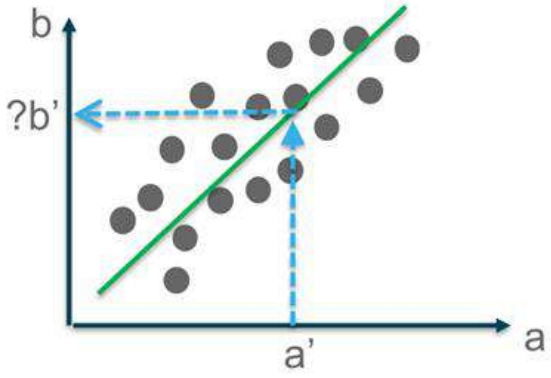
- **Features (independent variables)** — input data used for learning
- **Labels (dependent variables)** — outputs to be predicted (in supervised learning)

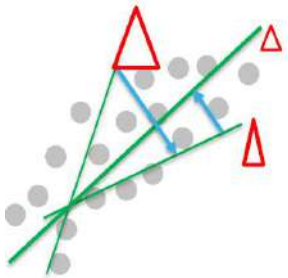
Two key stages:

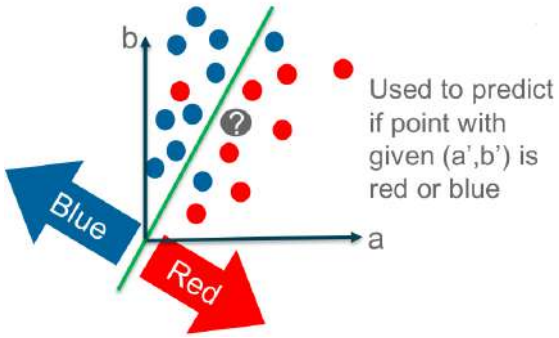
1. **Training** — build a model from data
2. **Inference (Prediction)** — apply the model to new data

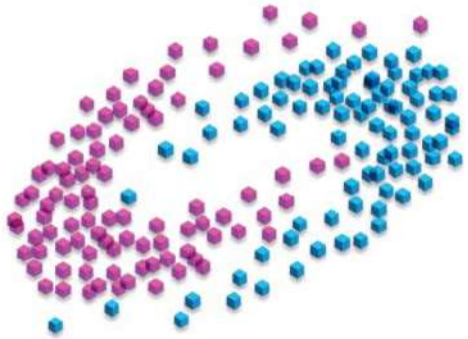
Basic tasks solved by ML

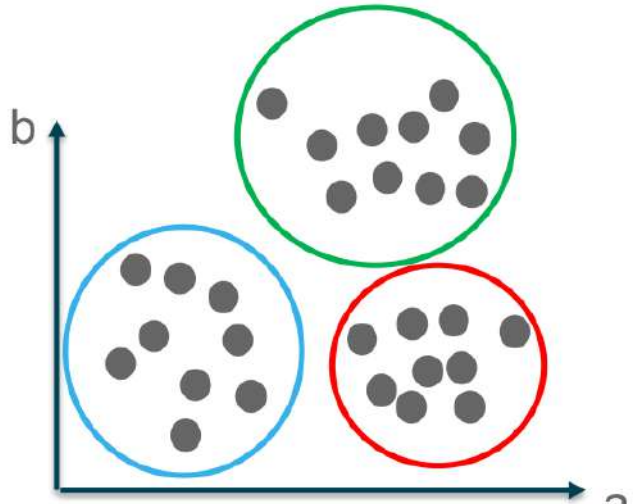
Regression **Classification** **Clustering** **Dimension reduction**

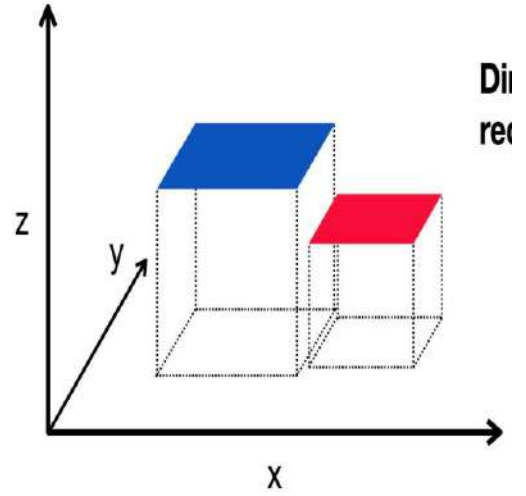


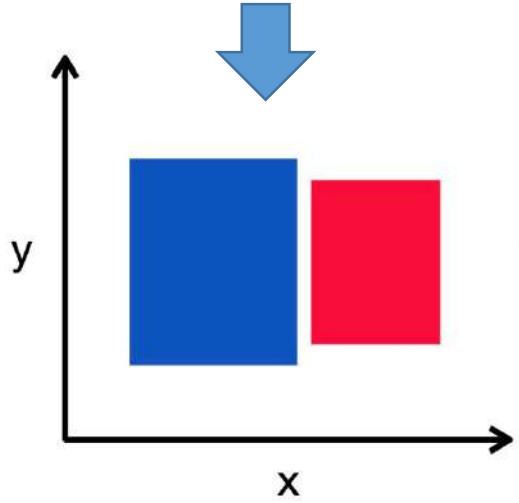




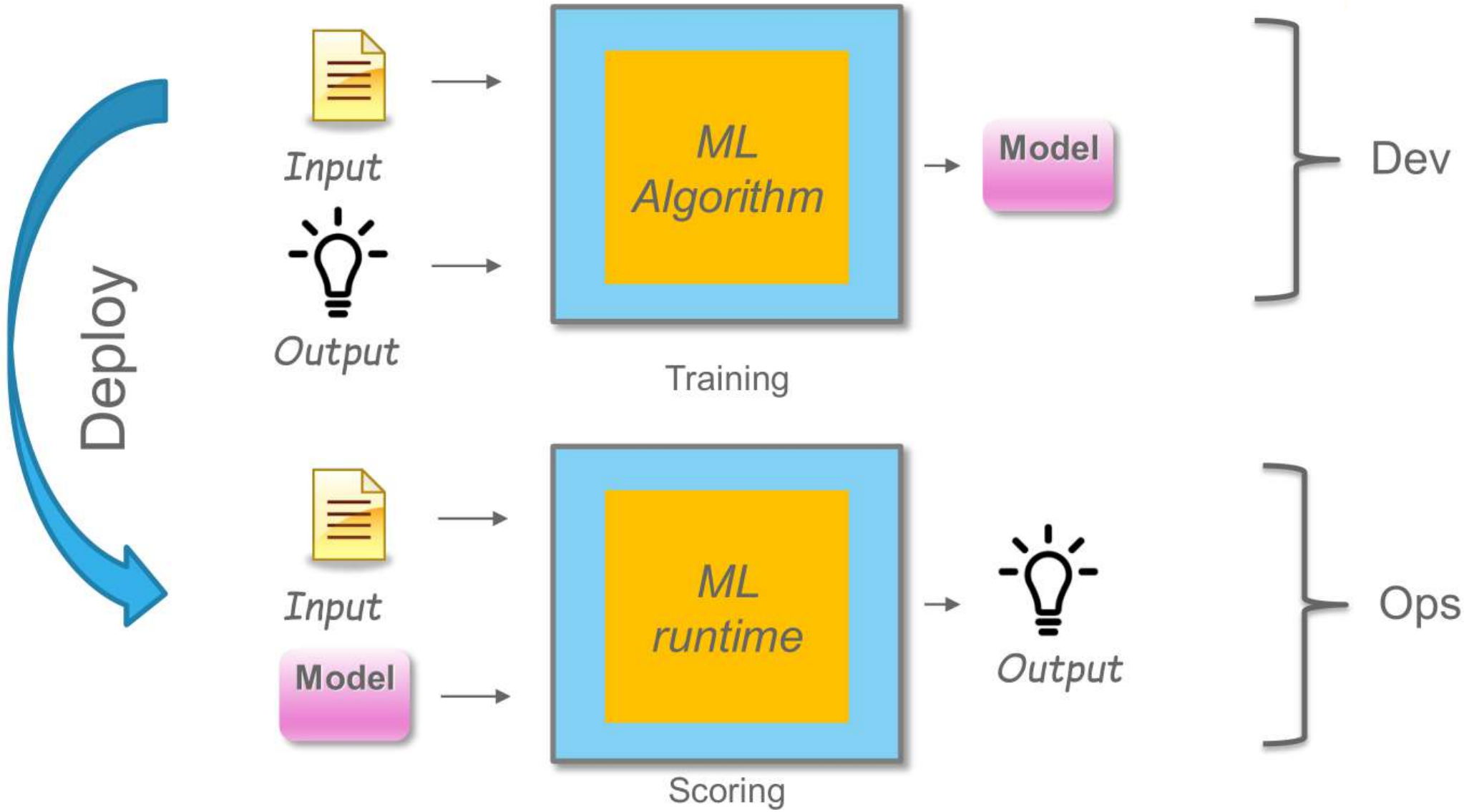




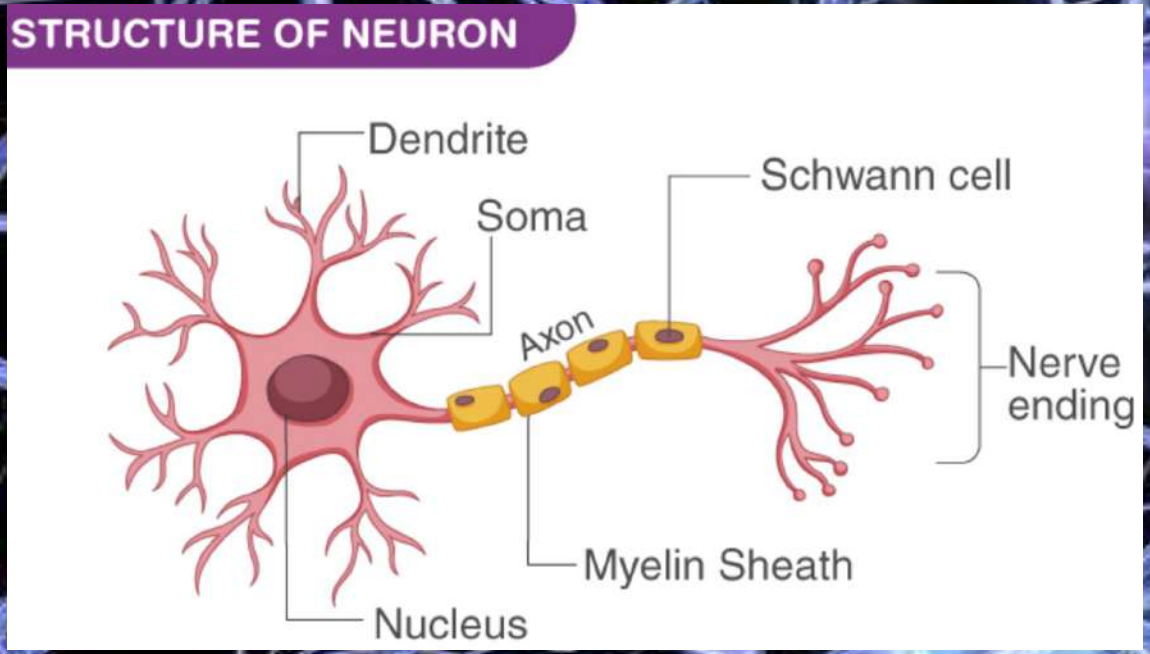


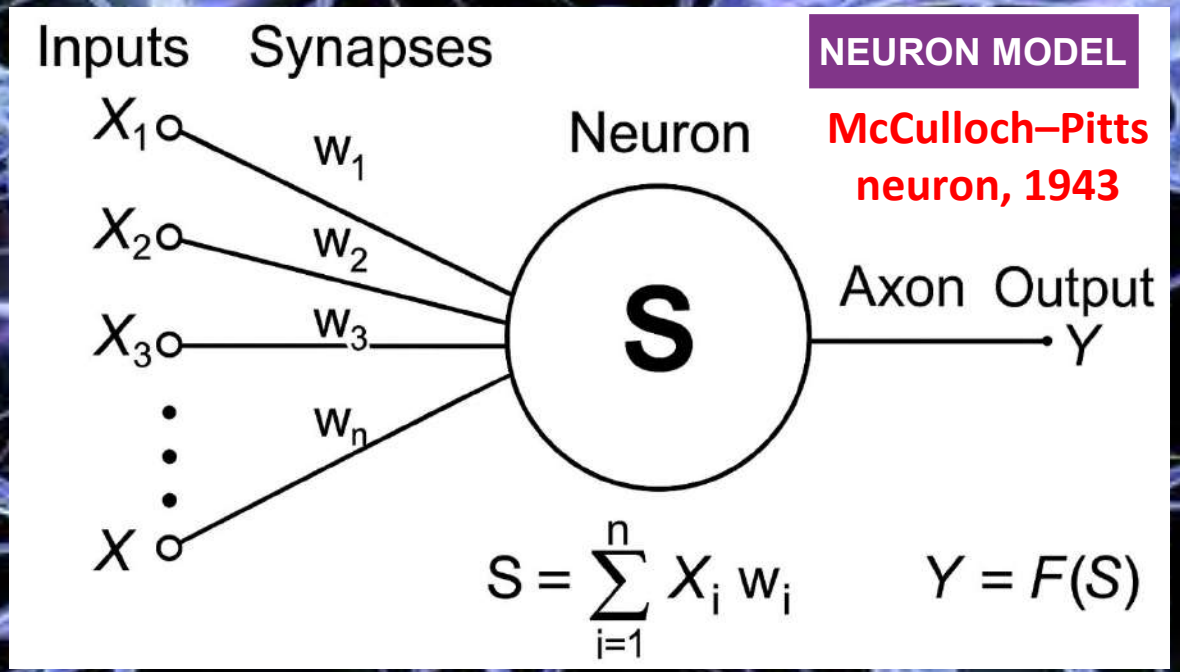


What is supervised Machine Learning?



Neural network





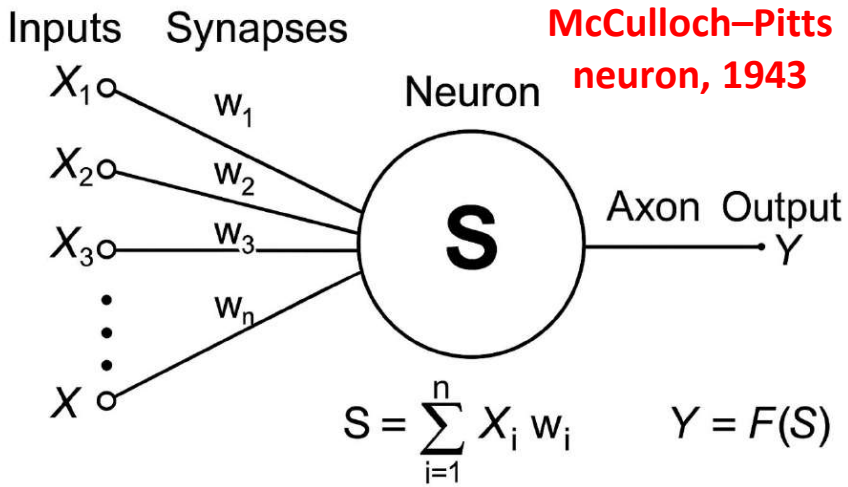
The term AI was inspired by the brain/human intelligence!

Artificial Neuron: Mimicking the Brain to Build Intelligence

Inspiration from Cybernetics

- The idea of **copying brain function** using simplified models of **neurons** dates back to the birth of **cybernetics** (mid-20th century).
- Even **without fully understanding** how real neurons work, scientists proposed models that imitate **key behaviors**.

How It Works:



McCulloch–Pitts neuron, 1943

- Inputs X_i are multiplied by **weights** w_i — like signals through synapses.
- The neuron **sums** these signals:

$$S = \sum_{i=1}^{n} X_i w_i$$

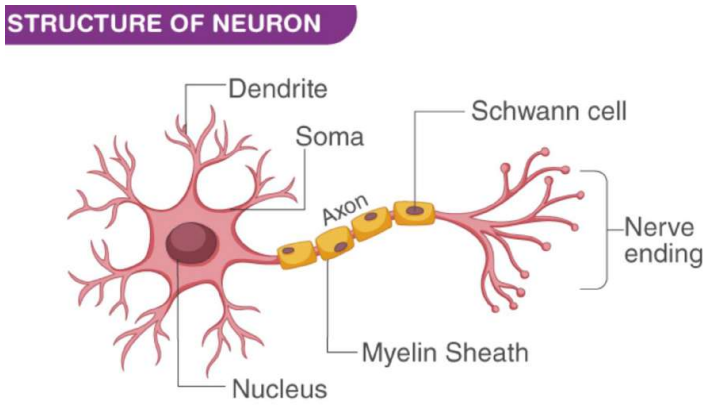
- Then applies a **nonlinear activation function** $Y = F(S)$ to decide whether to "fire".

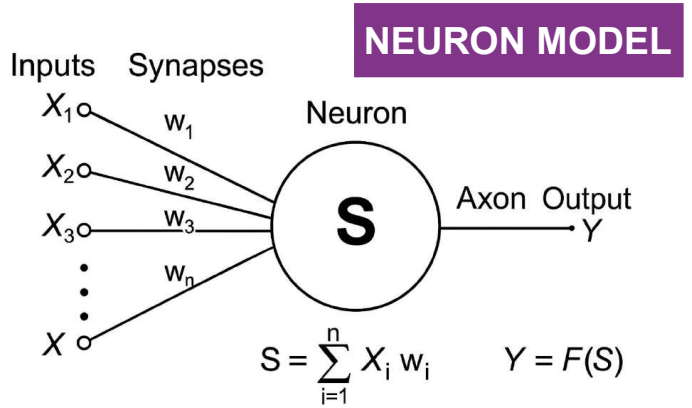
This simple model became the building block of **artificial neural networks** — the foundation of modern AI.

✅ **And it worked!**

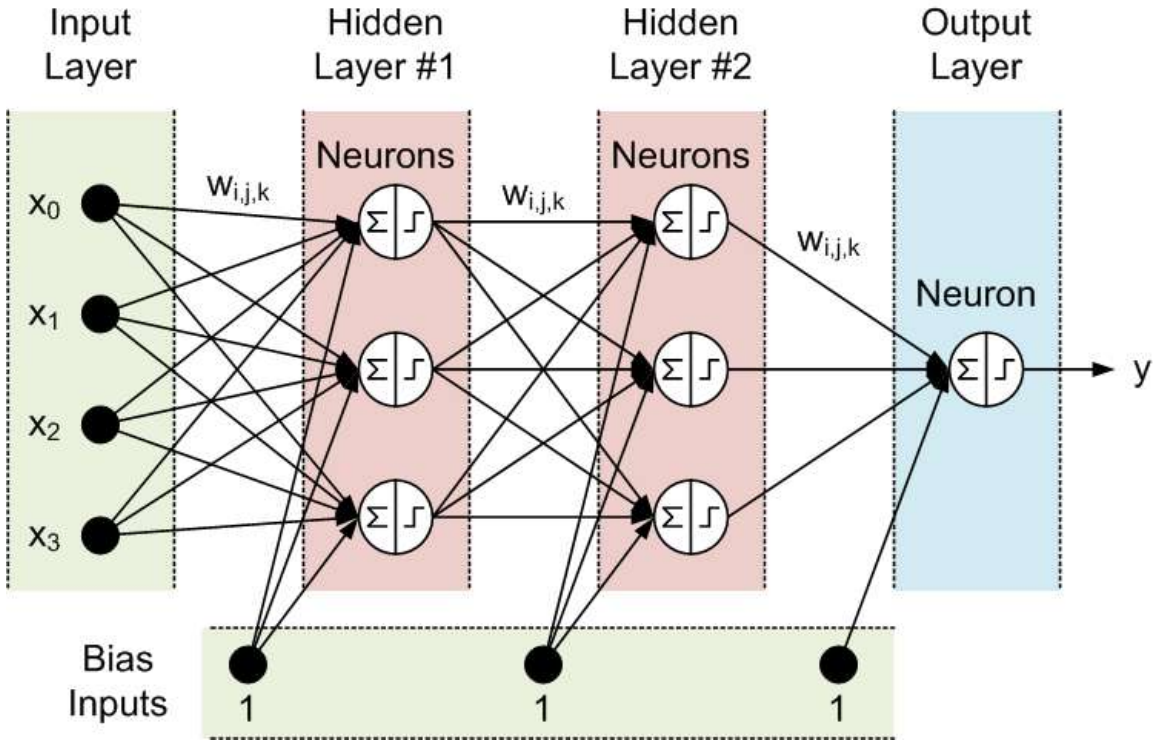
- Despite its simplicity, this approach led to systems that **recognize images, translate languages, and drive cars** — all by mimicking nature's logic, not copying its biology exactly.

Neural network



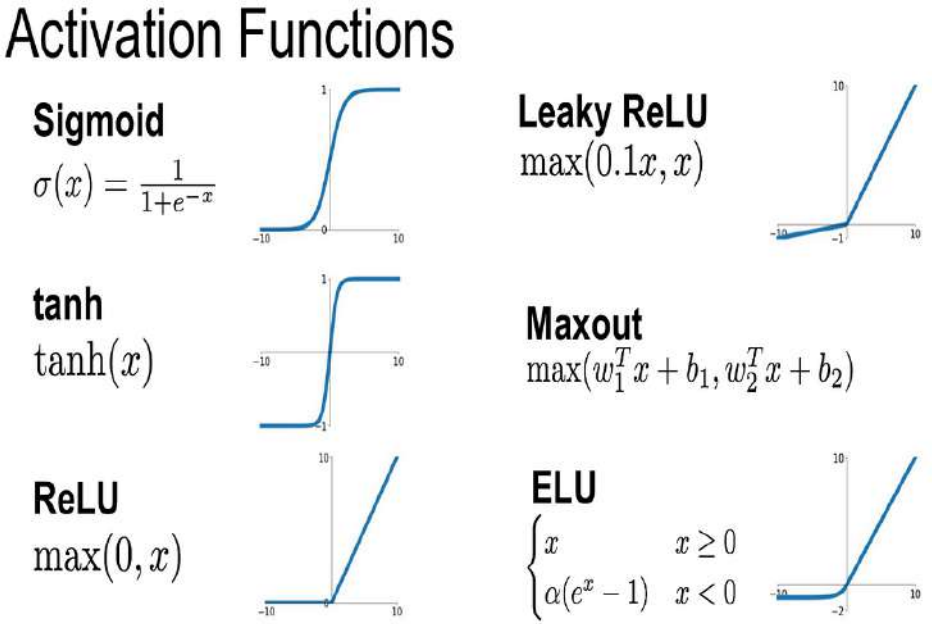


$$f(x_1, \ldots, x_n) = \mathbf{w}_0 + \mathbf{w}_1 x_1 + \ldots + \mathbf{w}_n x_n$$

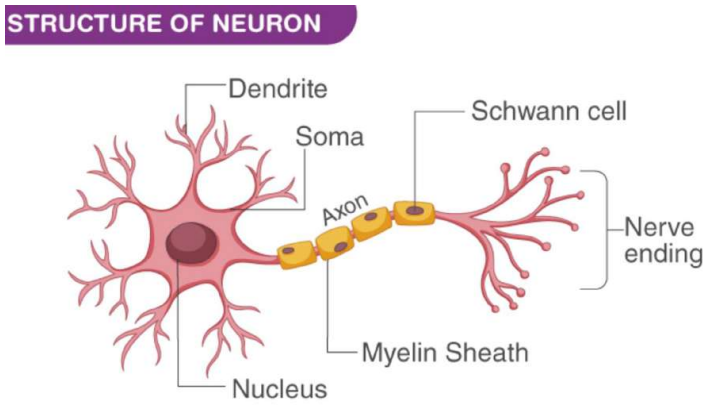


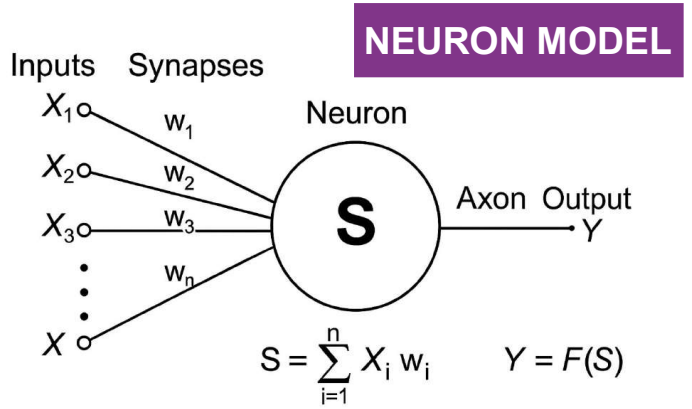
Types of relationships in neutrons

- ReLU
- Sigmoid
- TanH
- SoftMax

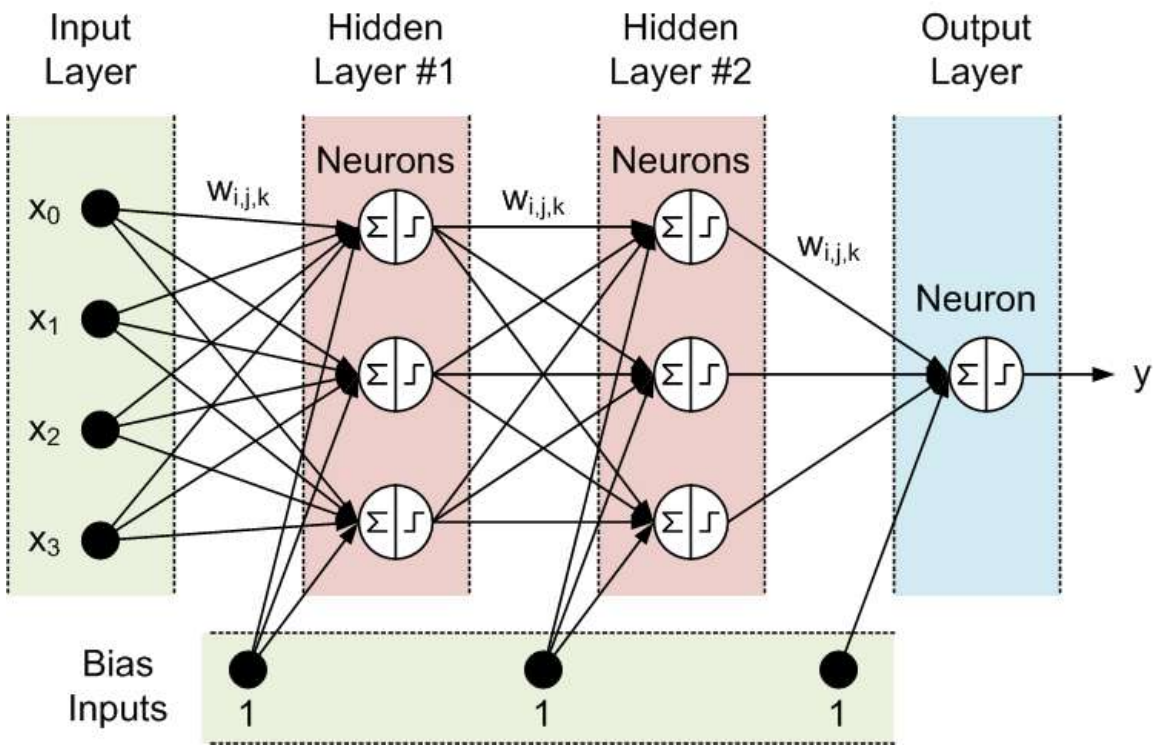


Neural network



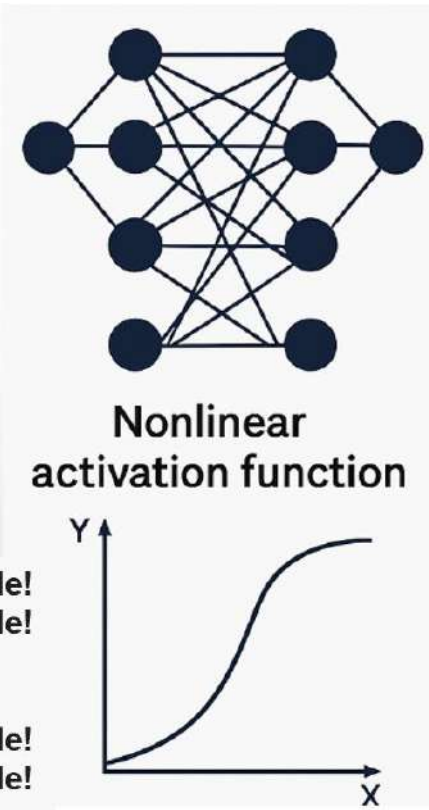


$$f(x_1, \ldots, x_n) = \mathbf{w}_0 + \mathbf{w}_1 x_1 + \ldots + \mathbf{w}_n x_n$$



Types of relationships in neutrons

- ReLU
- Sigmoid
- TanH
- SoftMax



- **linear activation function F(S) leads to a linear system of relationships, without added complexity!**
- **All the magic comes from nonlinearity in activation functions giving flexibility and expressive power.**

🧠 Why Nonlinearity Makes Neural Networks Powerful?

Linearity: The Crown of Classical Analysis

- In classical math, **linearity** is the ideal: elegant, predictable, and solvable.
- Linear models are easy to analyze and combine, but linear layers stacked together are still... **linear**.

🔍 **Linear systems can’t model complex decisions, context, or boundaries.**

The Role of Nonlinearity in Neural Networks

- **Nonlinear activation functions** break the linear deadlock.
- They allow:
  - Hierarchical abstraction
  - Conditional logic (if-then-like behavior)
  - Universal approximation (any function)

Nonlinearity makes neural networks truly *learn* — not just transform.

A Philosophical Analogy

Classical Physics vs. Quantum Mechanics

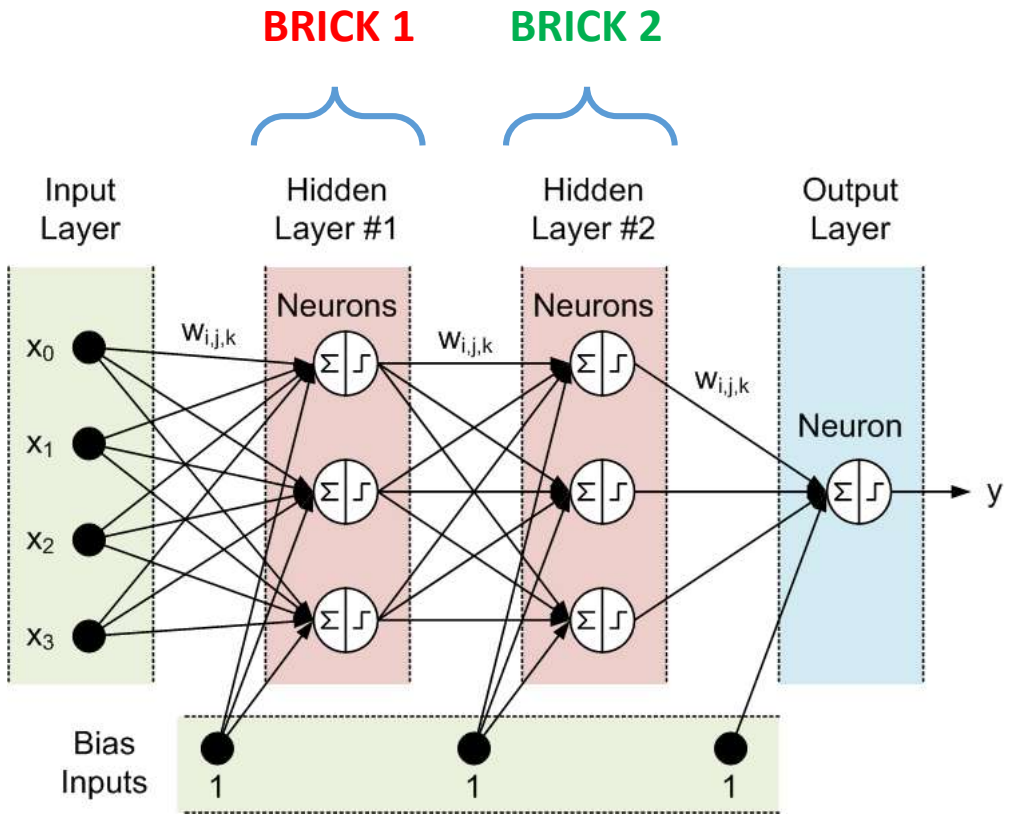
Classical Approach	Neural Networks / AI
Linear, deterministic models	Nonlinear, emergent behaviors
Full control and transparency	Black-box learning, self-adjusting
Rule-based deduction	Pattern-based induction

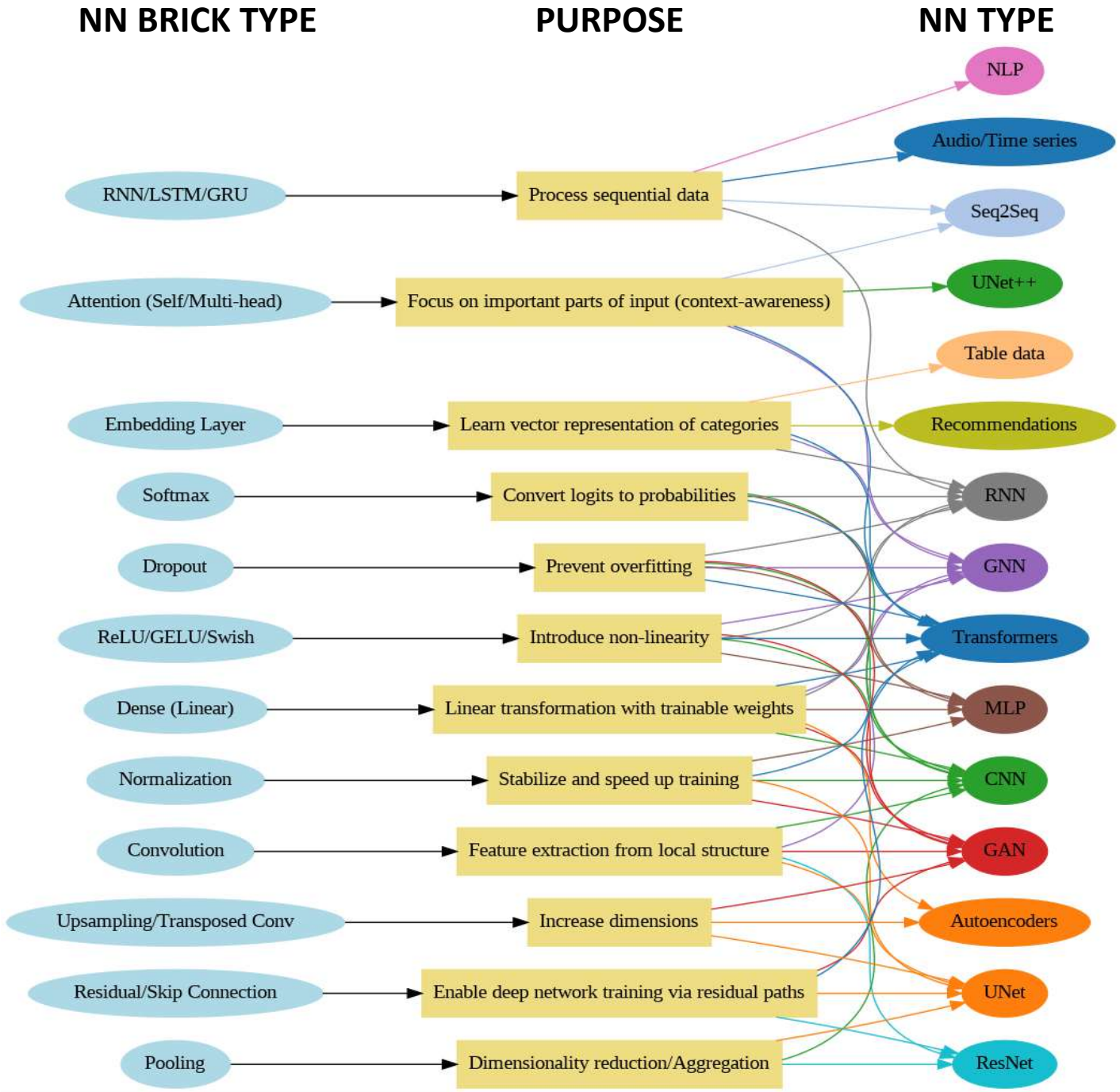
Deep Insight:

Linearity is the language humans use to formalize the simple.
Nonlinearity is the language nature uses to express the complex.

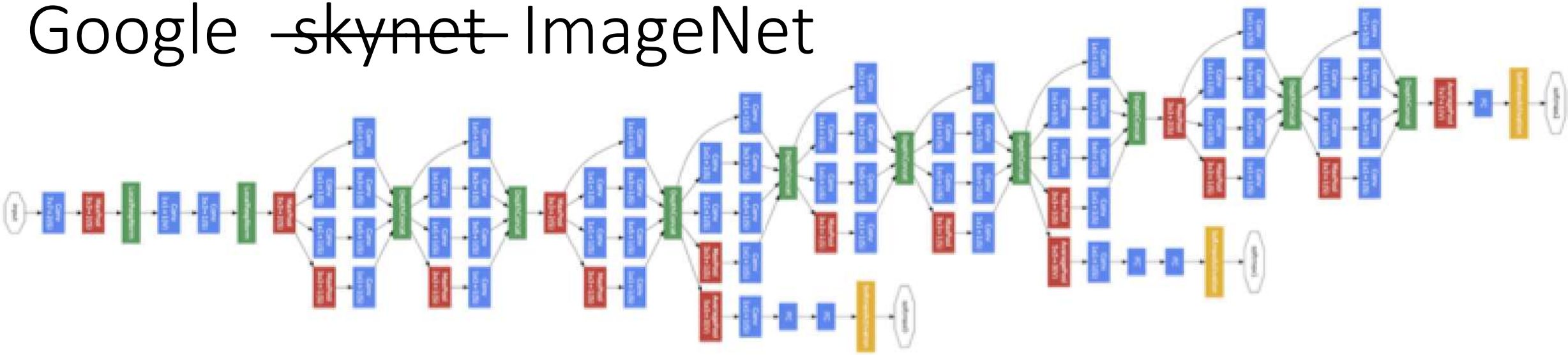
Neural networks succeed by embracing the **nonlinear nature of the real world**, just like modern data analysis had to.

Yet another brick in the wall… of NN



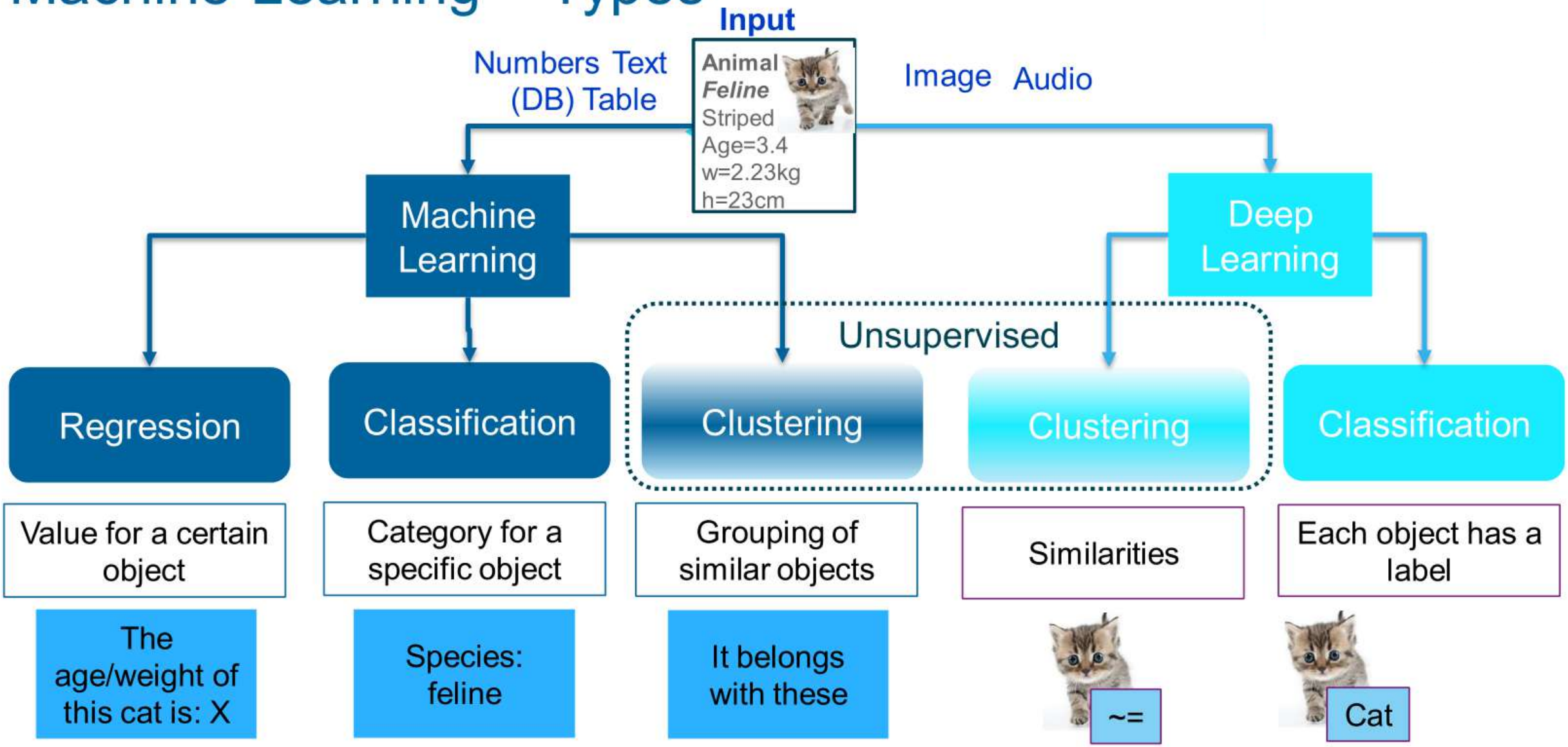


Google ~~skynet~~ ImageNet

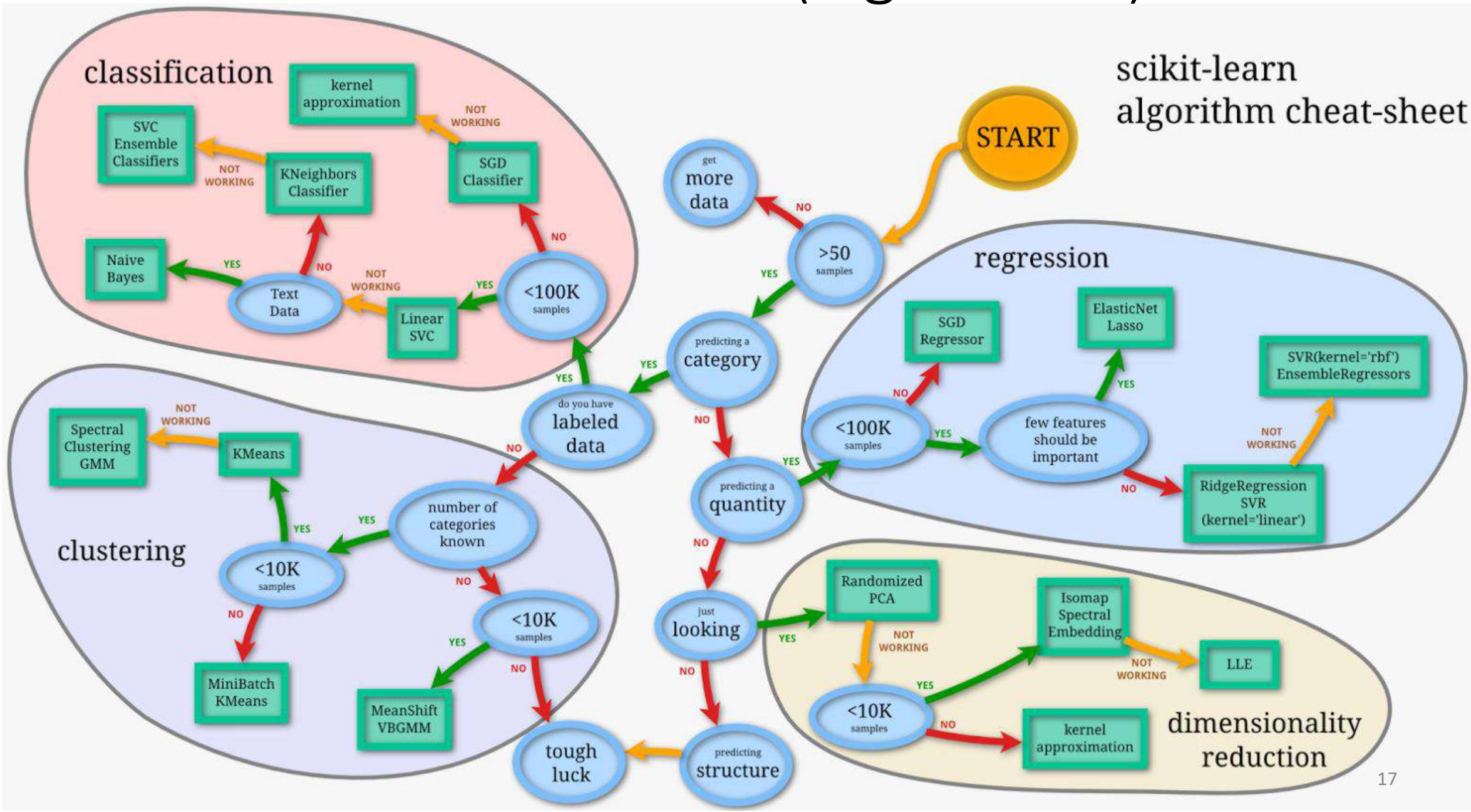


Convolution
Pooling
Softmax
Other

Machine Learning – Types



Zoo of ML-methods (algorithms)



Deep Learning (DL)

Deep Learning is a subfield of machine learning that uses **artificial neural networks with multiple layers** ("deep" architectures) to **automatically learn features** and make predictions from data. Inspired by the structure of the human brain, deep learning models can learn **complex patterns** from **large volumes of data** — especially effective for:

- Image recognition
- Natural language processing
- Speech recognition

Key Differences from Traditional ML:

Traditional ML	Deep Learning
Manual feature engineering	Learns features automatically
Requires structured data	Works well with unstructured data
Simpler models (e.g. SVM, trees)	Complex neural networks (CNN, RNN, etc.)
Performs well with less data	Needs large datasets and high compute

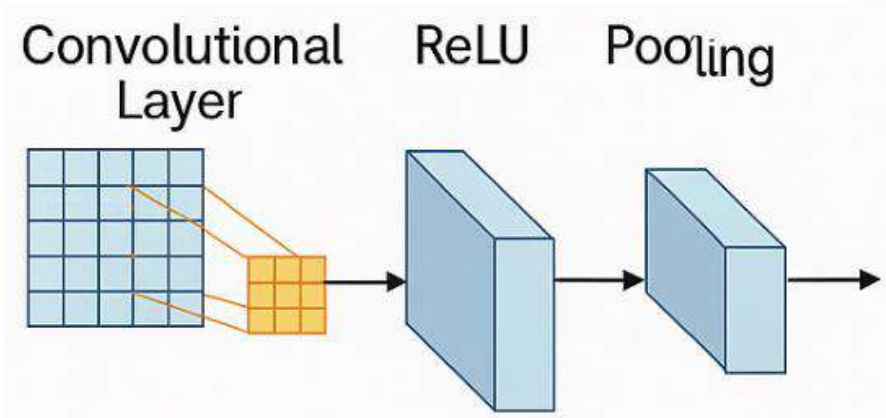
Convolutional Neural Networks (CNNs)

🔍 **Core Idea:** CNNs are designed to **automatically extract spatial features** from structured data like images, using the mathematical operation of **convolution**.

- Use **convolutional filters** to detect patterns (edges, textures, shapes)
- Apply **shared weights** across input space → fewer parameters
- Incorporate **pooling** to reduce dimensionality and focus on dominant features
- Often include **normalization** and **ReLU** for training stability and nonlinearity

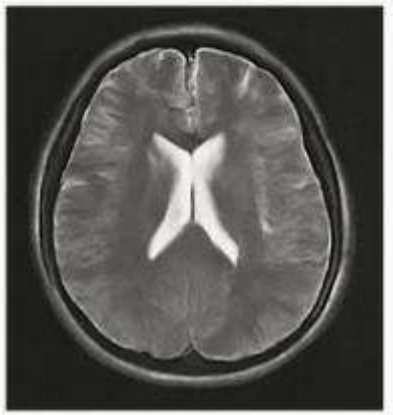
🧊 **Why CNNs Work So Well:**

- Local connections capture **spatial hierarchies** (e.g. from pixels to shapes)
- Translation invariance — detect features regardless of position
- Very effective at generalizing across large datasets (when trained properly)



Applications:

- 🖼️ **Image classification** (e.g. cats vs. dogs)
- 🔍 **Object detection** (e.g. YOLO, Faster R-CNN)
- 🧠 **Medical imaging** (MRI, CT scans, X-rays)
- 🚗 **Autonomous driving** (road/lane detection)
- 🛰️ **Remote sensing**, industrial inspection, face recognition, etc.



📌 **CNNs are the foundation of modern visual recognition systems.**

Computer Vision: A Leading AI Application

Computer vision is one of the most actively developed and widely applied areas of artificial intelligence.
Its goal is to enable machines to “see” and **interpret visual information** — just like humans do.

Why It’s So Popular:

- **Massive real-world demand** for visual understanding
- Combines deep learning, big data, and powerful hardware

Key Applications:

- **Robotics** – gives robots the ability to recognize and navigate their environment ("robot eyes")
- **Autonomous vehicles** – essential for object detection, road tracking, and safety
- **Healthcare** – used in medical imaging (X-rays, MRIs, pathology slides) for diagnosis and monitoring
- **Retail & Security** – facial recognition, shelf monitoring, customer behavior analysis
- **Agriculture & Industry** – plant health monitoring, defect detection, automation

As cameras become ubiquitous and visual data explodes, computer vision is central to many AI-powered systems.

Key Global Computer Vision Competitions

ImageNet Challenge (ILSVRC, 2010–2017)

- ImageNet was the milestone benchmark
- Made deep learning mainstream (AlexNet 2012 breakthrough)
- Still used for pretraining and benchmarking

CVPR Challenges (annual)

- Held at the top CV conference CVPR
- Topics: Object Detection, Video Analysis, 3D Vision, VQA, etc.
- Hosted by CVF (Computer Vision Foundation)
- Example: COCO Dataset & Object Detection Challenge

ECCV & ICCV Challenges

- ECCV (Europe) and ICCV (International)
- Similar themes: segmentation, recognition, scene understanding
- Often industry-sponsored (e.g. Google, Facebook AI)

Kaggle Competitions

- Open competitions with real-world datasets and cash prizes
- Example CV challenges (among 200+ available):
  - Diabetic retinopathy detection
  - Pneumonia classification from X-rays

📈 Trends and Legacy Challenges

⚙️ Modern Trends in CV Challenges

- 🔁 **Beyond classification**:
  - Object detection, instance segmentation, pose estimation
- 🧠 **Multimodal models**:
  - Vision + Language (e.g., VQA, captioning)
- 🌐 **Generative Vision**:
  - Diffusion models, GANs, image synthesis (see: DALL·E, Midjourney)

Computer vision competitions remain a driving force for innovation, benchmarking, and practical progress in AI.

🔁 Recurrent Neural Networks (RNNs)

🧠 **Core Idea:** RNNs are designed to **process sequential data** by maintaining an internal state (memory) that evolves over time.

- At each time step, the network takes an input and updates a hidden state based on the **current input and the previous state**
- This allows RNNs to model **temporal dependencies** and **ordered information**
- Useful when context and order matter

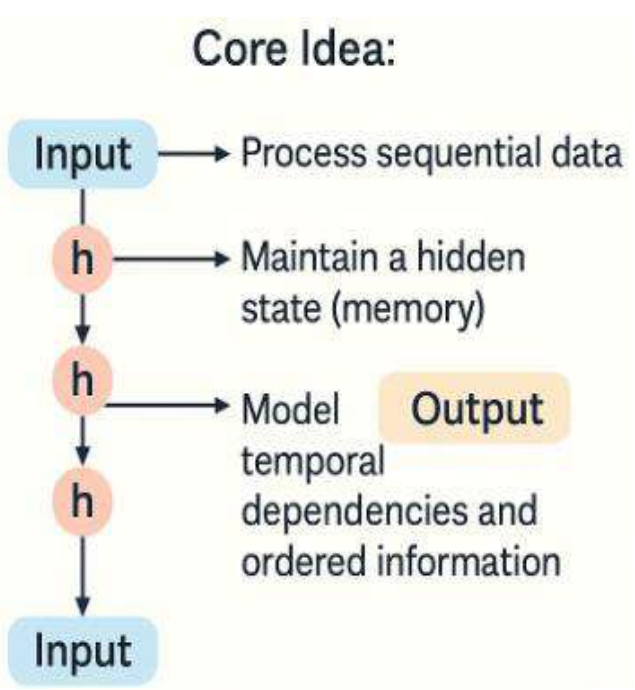
⏳ **How They Work:**

- Share weights across time steps (unlike CNNs across space)
- Can "remember" past inputs — theoretically unlimited sequence length
- Trained using **Backpropagation Through Time (BPTT)**

Applications:

- 📝 **Natural Language Processing (NLP)** – sentiment analysis, translation, text generation
- 🎶 **Speech recognition & audio modeling**
- 📈 **Time series forecasting** – stock prices, weather
- **Biological sequence analysis** (DNA, protein sequences)

📌 RNNs capture **context** in data where **order matters** — unlike CNNs which treat input more spatially.



Transformers

Attention Is All You Need

Core Idea: Transformers are deep learning architectures designed to **model relationships in sequences** using **attention mechanisms** — without relying on recurrence.

- Replace recurrence (RNNs) with **self-attention**
- Each input element can attend to **any other** in the sequence
- Allows **parallel computation** and better **long-range dependency modeling**

Key Components:

- **Self-Attention**: dynamically weighs relevance between tokens
- **Multi-head Attention**: multiple attention "views" capture diverse patterns
- **Positional Encoding**: adds order-awareness without recurrence
- **Feedforward layers** and **Layer Normalization**

Applications:

- **Natural Language Processing (NLP)** – translation, summarization, chatbots
- **Large Language Models** – GPT, BERT, T5
- **Music generation**, **time series forecasting**, **vision transformers (ViT)**

Transformers revolutionized AI by enabling **scalable learning on sequences** — across text, images, and more.

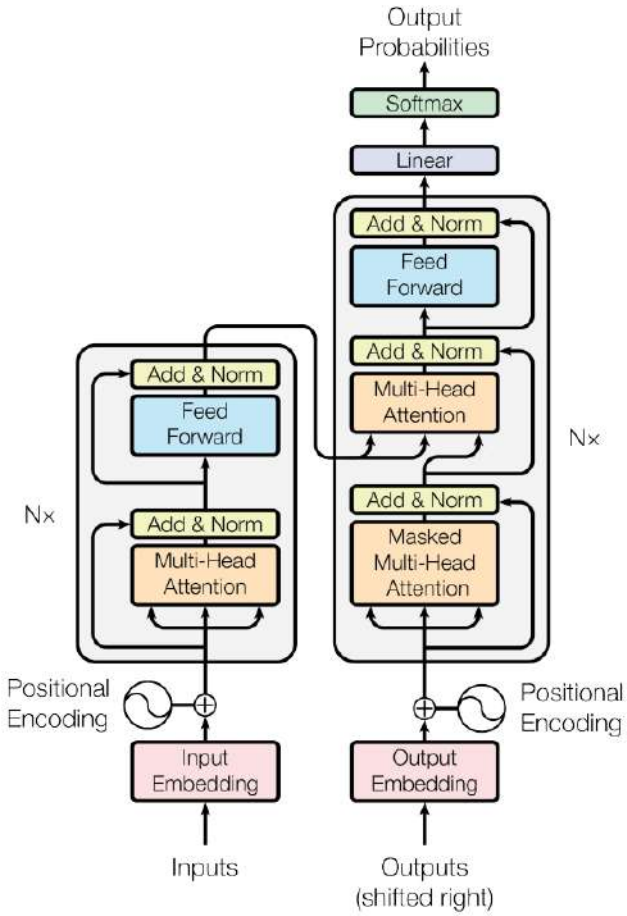


Figure 1: The Transformer - model architecture.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

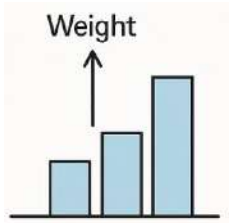
$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Attention in Transformers & the Boltzmann Distribution: An Analogy

🎯 **Concept:** both formulas compute **probability distributions**, assigning higher weight to more "relevant" or "favorable" elements.

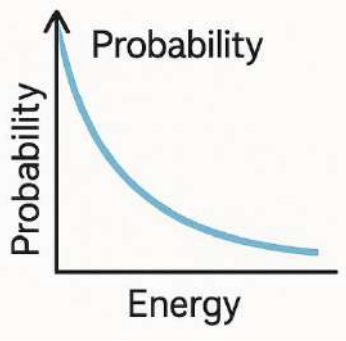
Attention in transformer:

$$\mathrm{w_i} = \mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} = \frac{e^{x_i/\mathbf{T}}}{\sum_j e^{x_j/\mathbf{T}}}$$



Probability over system energy:

$$P(E_i) = \frac{e^{-E_i/(kT)}}{\sum_j e^{-E_j/(kT)}}$$



Component	Attention (AI)	Boltzmann (Physics)
What is distributed?	Attention across tokens	Probability over energy states
High value means	More attention	Lower energy → higher probability
Exponential scaling	Highlights differences in relevance	Highlights differences in energy
Temperature TTT	Used in softmax with temperature	Controls sharpness of distribution

🔥 Temperature in Attention

In AI, **temperature T can be added as parameter.** Low T → sharper focus, more deterministic results. High T → more uniform attention, more uncertainty in the results. **Takeaway:** the attention mechanism in Transformers works on the **same principles** as thermal energy distributions: **Tokens with high similarity are like states with low energy — they become more likely**.

Energy (PHYS) is the negative proximity between words (AI).

Attention and Energy: A Philosophical Analogy

In Physics:

The universe seeks rest.

Systems **prefer low energy** — quiet, stable, natural.

Probability flows toward **minimal resistance**, governed by the law of **least action**.

$$P(E) \propto e^{-E/kT}$$

Key Insight:

Energy is the inverse of semantic affinity.

Where physics moves away from effort, **attention moves toward relevance.**

In Attention:

The mind seeks meaning.

Words **prefer closeness** — relevance, connection, sense.

Probability flows toward **maximum alignment**, guided by the law of **contextual attraction**.

$$P(x) \propto e^{x/T}$$

Interpretation:

- **Softmax is our modern Boltzmann.**
- Instead of particles and heat, we have **tokens and meaning**.
- The transformer doesn't just compute — it **attends**,
  seeking **clarity** the same way nature seeks **balance**.

Nobel Prize in Physics 2024: AI Meets Physics

🔬 **Awarded To:**

John J. Hopfield and **Geoffrey Hinton**

For pioneering work applying **physical principles** to **neural networks and machine learning**.

🧠 **Their Contributions:**

- **Hopfield Networks (1982):**
  Memory modeled as **energy minimization** — inspired by spin systems in statistical physics.
- **Boltzmann Machines (Hinton):**
  Learning via **thermal sampling** and **energy-based probability** (Boltzmann distribution).

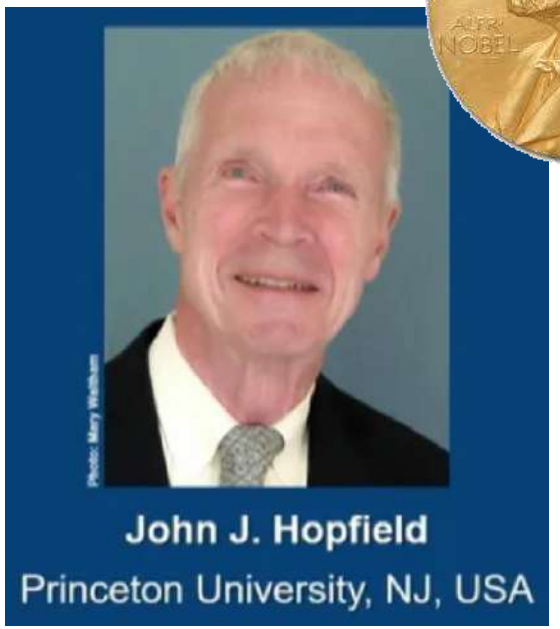
⚛️ **Why It's Physics:**

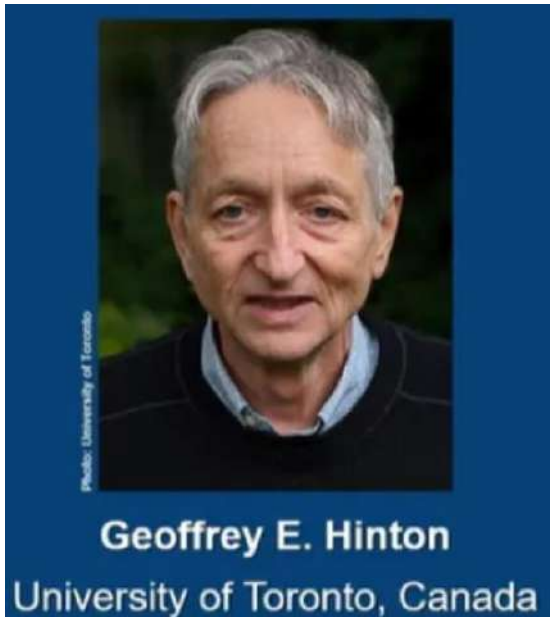
- Used concepts from **thermodynamics, energy landscapes, equilibrium**
- Neural computation viewed as a **physical system** evolving toward stable states

🌍 **Impact:**

- Foundations for **deep learning**, **generative models**, and **energy-based AI**
- Bridged the gap between **natural laws** and **artificial intelligence**

"They didn't just teach machines to learn —
they taught us that **learning is a physical process**."





🚀 Why Attention in Transformers is More Efficient than RNNs

🧠 1. Global View vs. Sequential Processing

- **RNNs** process input **word by word**, step by step.
- **Attention** looks at the **entire sequence at once**.
- This allows each word to directly focus on all others, regardless of position.

⚡ 2. Parallelization

- **RNNs** are inherently **sequential** → no parallelism.
- **Attention** allows **parallel computation** across all positions.
- 🚀 Enables **faster training and inference** on GPUs.

🧵 3. Long-Range Dependencies

- In **RNNs**, signals from distant words fade over time.
- In **Attention**, each word can directly attend to any other → stronger connections.

🔧 4. Simpler Architecture

- **RNNs** rely on complex gates and memory (e.g., LSTM, GRU).
- **Attention** uses basic matrix operations + softmax → easier to implement and scale.

🔍 5. Interpretability

- Attention can be **visualized** to show what the model is focusing on.
- RNNs have opaque hidden states that are hard to interpret.

Feature	RNN	Transformer (Attention)
Processing	Sequential	Parallel
Long-range context	Weak	Strong
Speed	Slower	Much faster
Architecture	Complex	Simple & efficient
Interpretability	Low	High

🧠 Natural Language Processing (NLP): The Second Pillar of Modern AI (After CV)

📊 NLP vs. CV — Twin Giants of AI

- After the success of **Computer Vision NLP** became the **second dominant AI field**, and now often **competes** in scale and impact.
- Why? Because **most real-world big data is textual**! emails, chats, documents, web pages, scientific papers, code...

💼 Commercial Momentum

- NLP powers **search engines, assistants, chatbots, translation**, and now **generative AI (LLMs)**.
- Text data is **plentiful, labeled, and monetizable**.

🔬 NLP in Physics: A Late Bloomer

- In physics, **structured numerical data** dominates, not natural language.
- **Transformers** and NLP models struggle with **raw numbers** and **continuous values**.

❓ Challenges in Physics

- **How to encode numerical physical quantities** (e.g., time series, spectra)?
- **What does "similarity" mean** between physical states or measurements?
- Traditional NLP similarity (word meaning) ≠ physical similarity (e.g., energy states)

🔄 The Shift Has Begun

- New encoding methods (Fourier, positional, learned embeddings) are emerging.
- **Transformers are being adapted** for particle physics, astrophysics, materials science.

“NLP tools were born for words — but they’re learning to speak physics.”

GNN – Graph Neural Network

A **Graph Neural Network** is a neural model that learns from **graphs** — structures made of **nodes** and **edges** (e.g., social networks, molecules, citation graphs).

Core Idea:

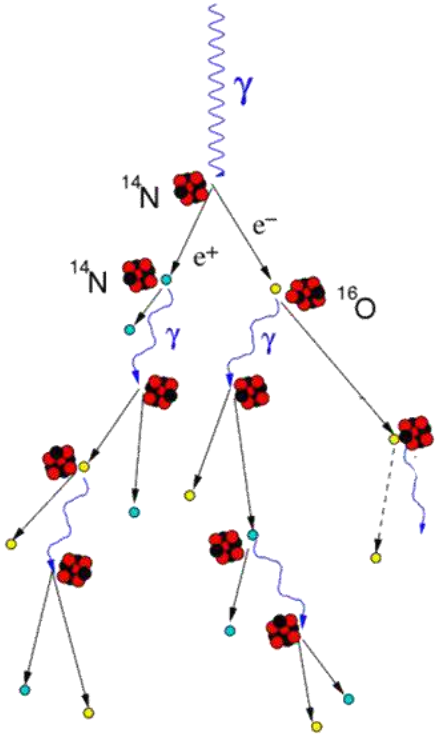
- **Each node updates its state** by aggregating information from its **neighbors**.
- Learning happens **not just from features**, but also from **structure**.

Basic GNN Workflow:

1. **Message Passing**:
   Nodes exchange information with neighbors.
2. **Aggregation**:
   Each node **combines** incoming messages (e.g., sum, mean, attention).
3. **Update**:
   Node updates its representation based on aggregated info.

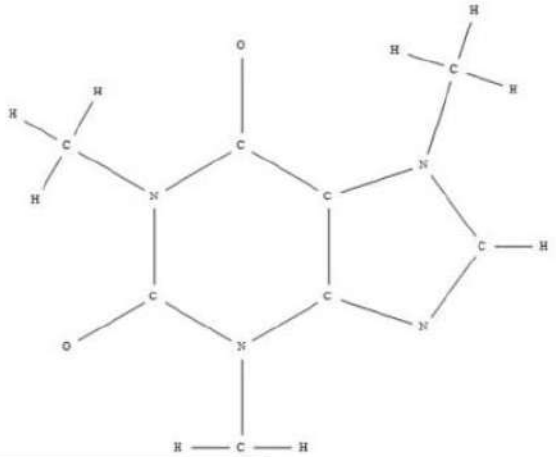
Why Use GNNs?

- Handles **non-Euclidean data**
- Great for **networks**, **recommendation systems**, **chemistry**, and **knowledge graphs**
- Models **relationships, influence, and structure** directly.

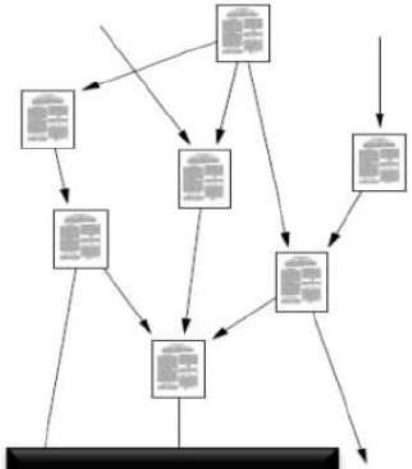


HE particle shower looks like graph!
⇒ GNN in action!

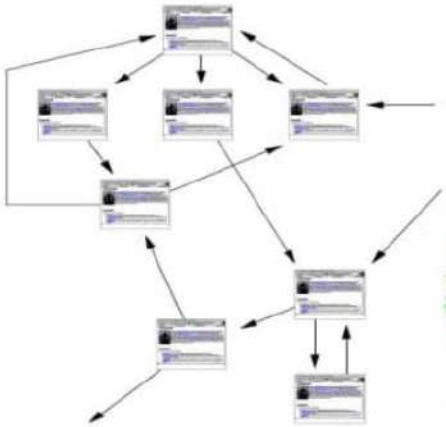
GNN applications



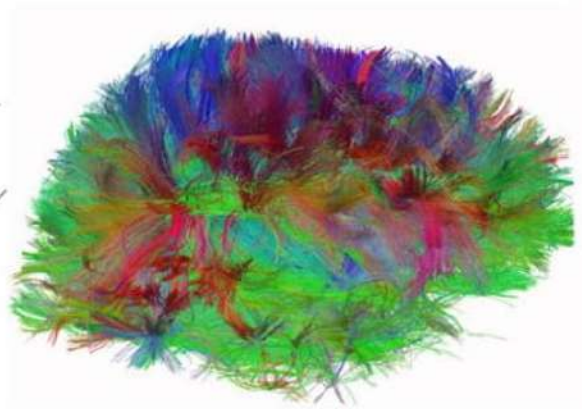
Molecules



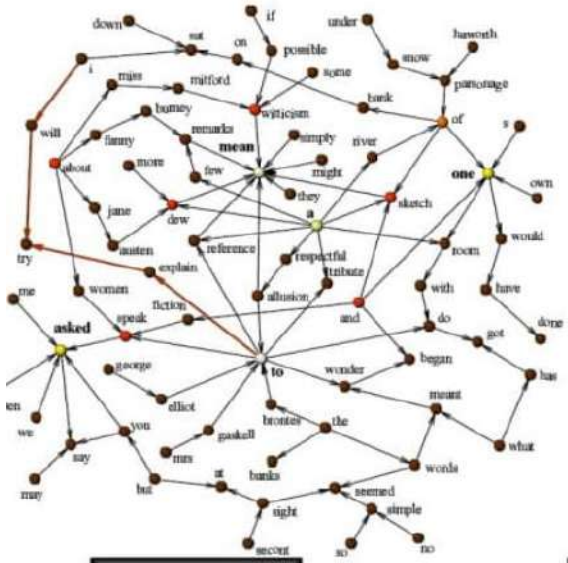
Knowledge



Information



Brain/neurons

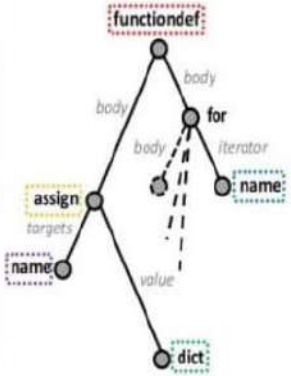


Genes

Communication

```
def encode(obj):
    """
    Encode a (possibly nested)
    dictionary containing complex values
    into a form that can be serialized
    using JSON.
    """
    e = {}
    for key,value in obj.items():
        if isinstance(value,dict):
            e[key] = encode(value)
        elif isinstance(value,complex):
            e[key] = {'type' : 'complex',
                      're' : value.real,
                      'i' : value.imag}
    return e

import ast
tree = ast.parse("...")
...
```



Software

Social

Generative Adversarial Networks (GAN)

What is a GAN? A GAN consists of two neural networks that compete with each other:

- The **Generator** tries to create fake data (e.g., images, detector signals)
- The **Discriminator** tries to tell real data from fake

How it works:

1. Generator takes random noise as input
2. Produces fake samples
3. Discriminator receives real and fake samples
4. Learns to classify real vs fake
5. Generator learns to fool the discriminator

Applications in Physics:

- Fast simulation (e.g., calorimeter showers — CaloGAN)
- Background generation
- Synthetic training data for rare processes

Pros:

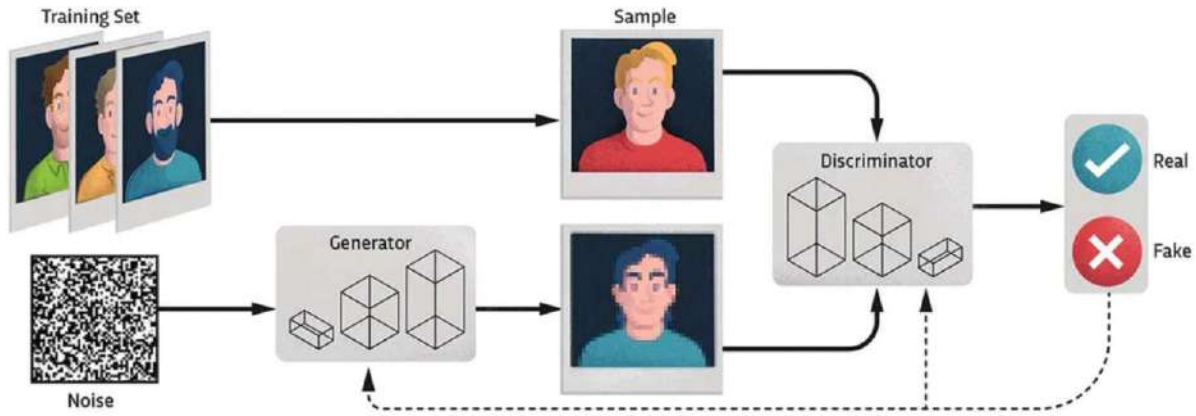
✔ Fast generation

✔ High visual realism

Cons:

⚠ Hard to train (unstable)

⚠ May generate unrealistic artifacts



≋ Diffusion Models – A New Generation of Generators

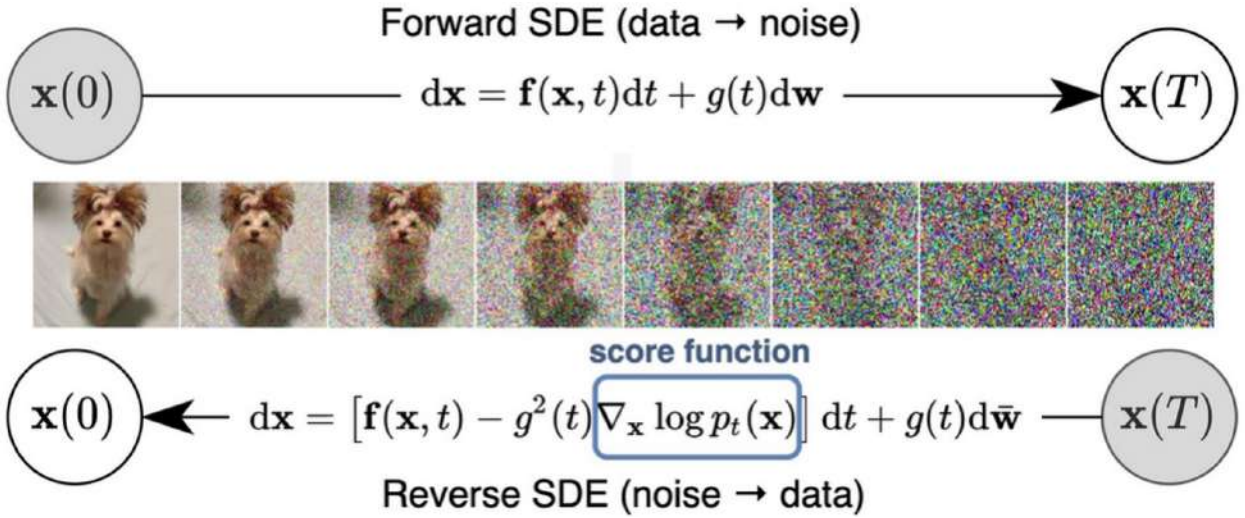
What is a Diffusion Model?

A generative model that learns to reverse a noise process:

- **Trained to denoise** step by step, turning random noise into realistic data

How it works:

1. Start with real data → gradually add noise (forward process)
2. Learn how to reverse the noise → recover clean data (reverse process)
3. At generation time: start from noise → apply reverse steps → get new sample



Applications in Physics:

- High-fidelity simulation of detector responses
- Background generation (e.g., DraPES model for dijet resonances)
- Event synthesis conditioned on physical variables

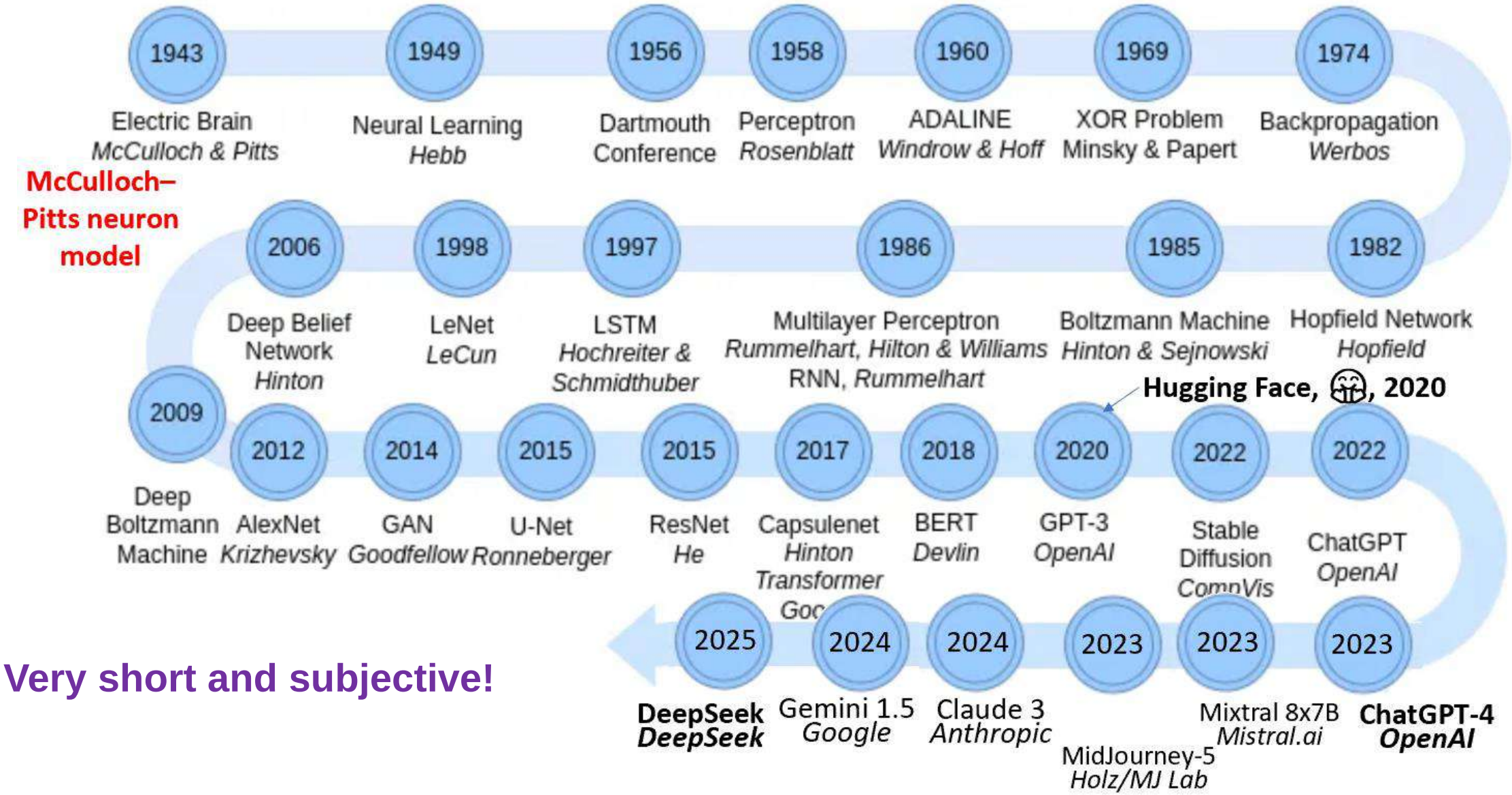
Pros:

✔ Stable training
✔ Excellent sample diversity and detail
✔ Maintains physical correlations

Cons:

⚠ Slower generation (many steps)
⚠ Computationally intensive

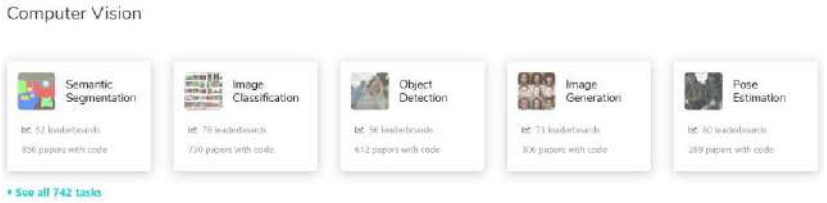
80+ year NN history

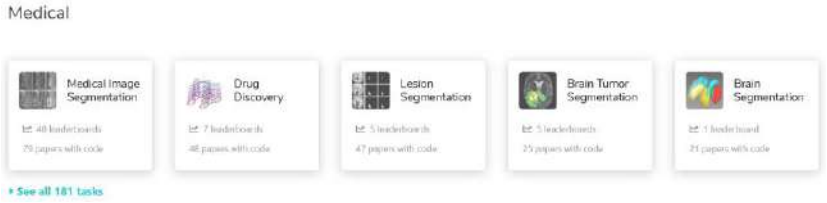


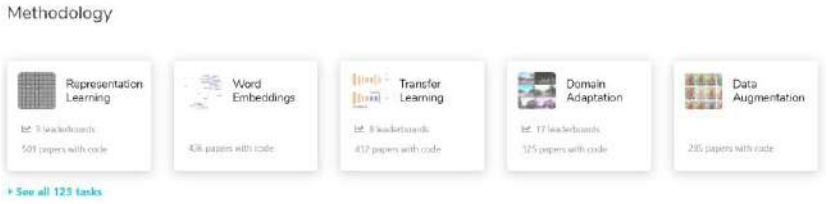
Very short and subjective!

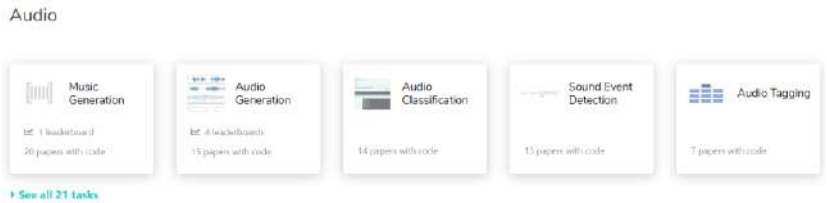
AI application zoo

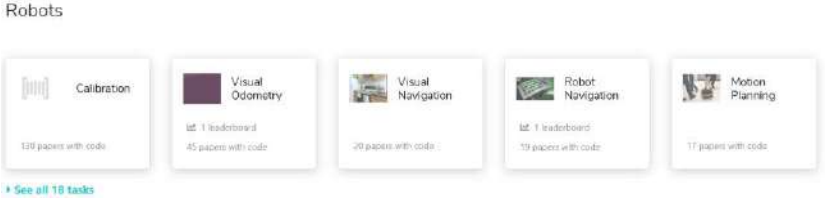
https://paperswithcode.com/sota

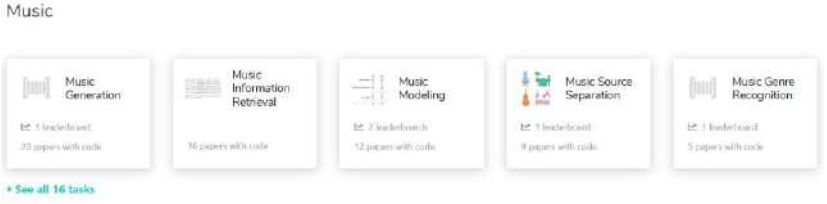


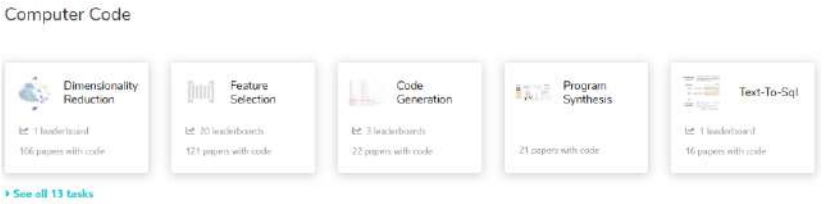


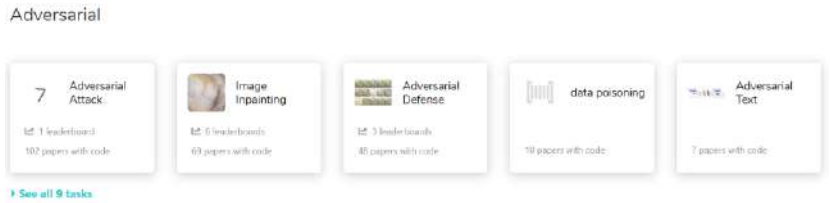


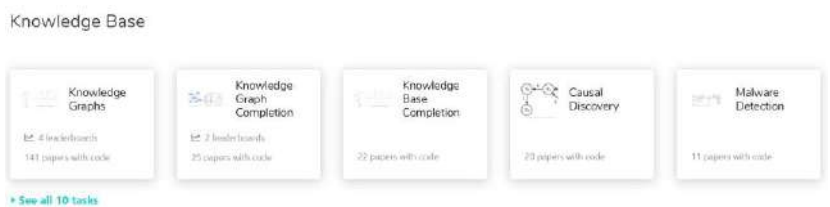












🔬 Applications in High-Energy Physics (HEP)

Goal: Show how different AI methods are solving real, high-stakes challenges in particle physics experiments like CMS and ATLAS at the LHC.

1. Event Classification: Jet Tagging

The problem: At the LHC, proton collisions create "jets" — sprays of particles. Physicists need to tell **which kind of particle** created the jet: was it a top quark, a gluon, a Higgs decay, or just background from QCD?

AI solution: Turn the jet into a machine-readable format and let neural networks classify it.

Representations of jets:

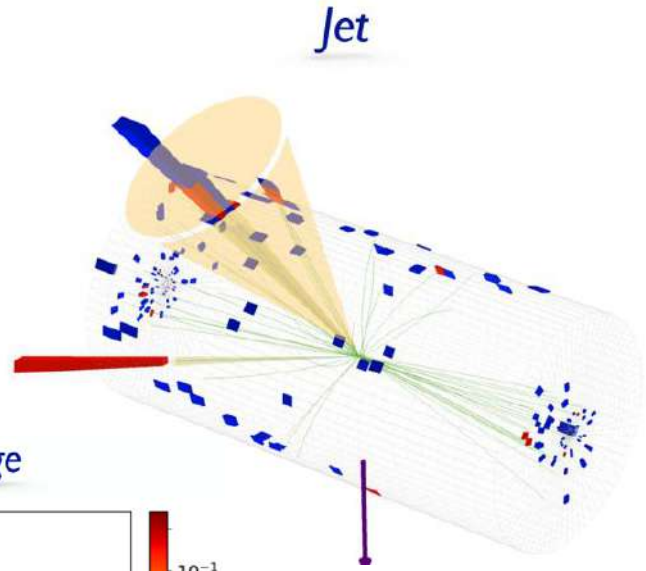
1. **Jet image** → use **CNNs** (e.g., ImageTop)
2. **Jet sequence** → use **Transformers**
3. **Jet particle cloud** → use **GNNs** (e.g., **ParticleNet**)

Example architectures:

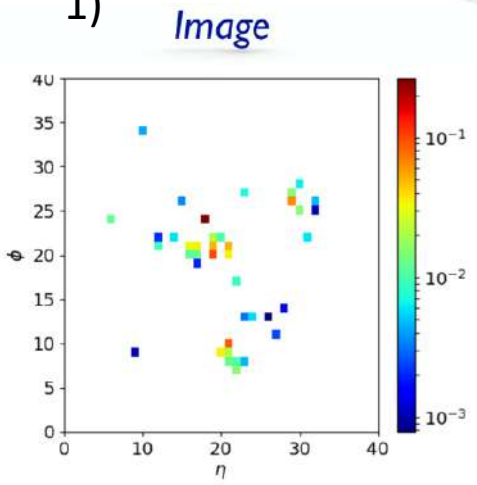
- **DeepCSV** / **DeepJet** (CMS): DNNs using high-level features for b-tagging
- **ParticleNet**: GNN on particle clouds; state-of-the-art for top/Higgs jets
- **DeepAK8**: detects Higgs → cc decays in boosted jets
- **PFN**: lightweight model using unordered particle inputs

Why AI is better than classical methods:

- Higher classification accuracy
- Learns subtle patterns that physicists didn't explicitly define
- Can separate hard-to-distinguish cases (e.g., quark vs gluon jets)

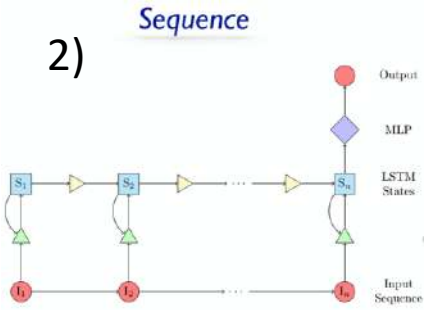


1)



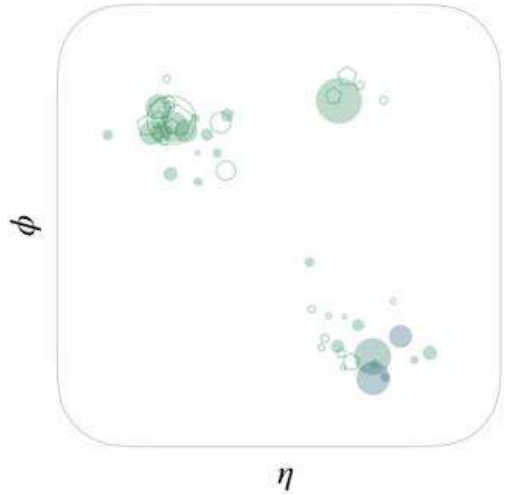
e.g., review in Kagan, arXiv:2012.09719

2)



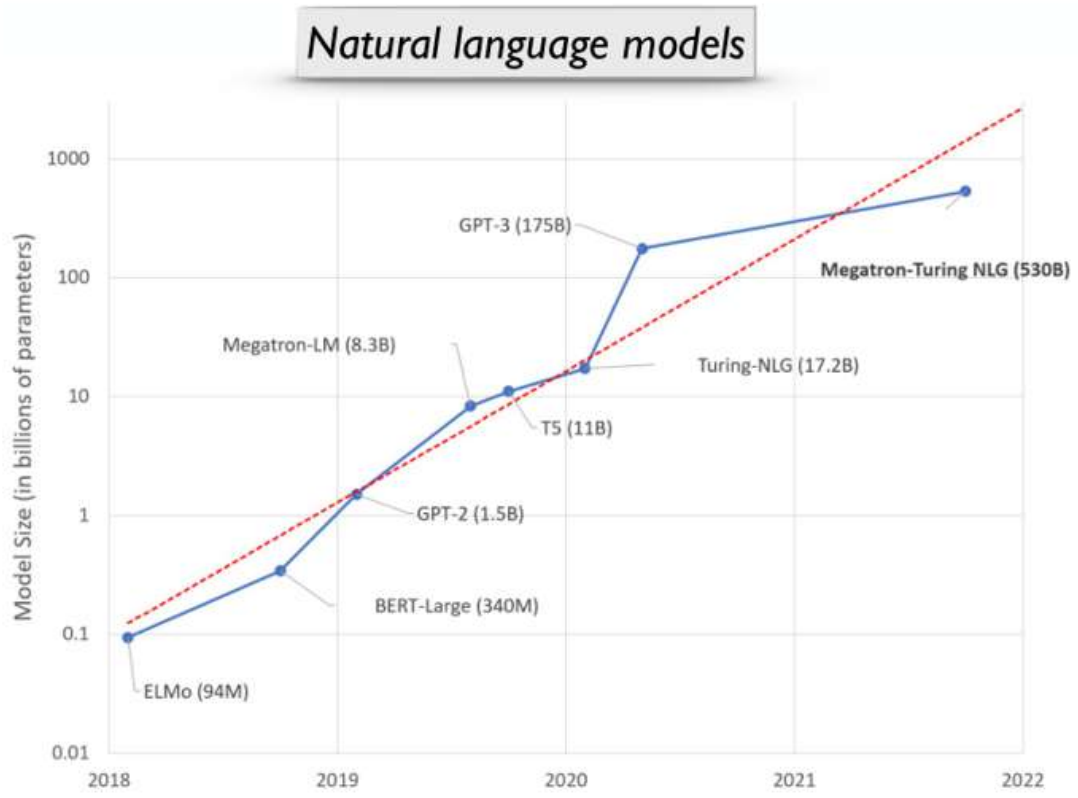
e.g., Guest, Collado, Baldi, Hsu, Urban, Whiteson arXiv: 1607.08633

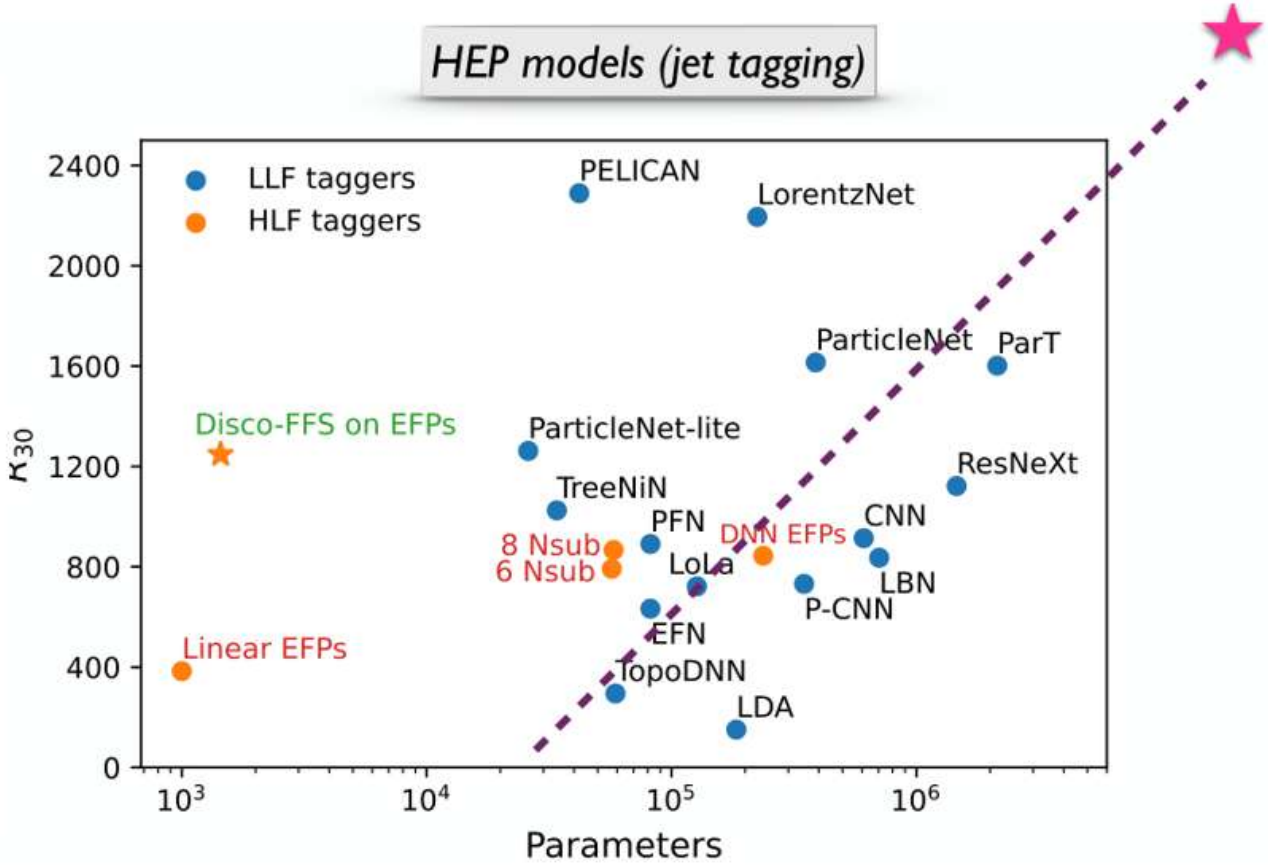
3)



Data coding matters!

Large Language Models: A New Moore's Law?





The number of parameters (and size) of models grows exponentially?

▼ Side note: Mass sculpting and decorrelation

Sometimes neural networks "cheat" — they use jet mass to make a decision.
But if you later want to study mass distributions, this creates **bias**.

✅ Solutions:

- **Adversarial training** (DeepAK8): forces network to ignore mass
- **Training on variable mass signal** (ParticleNet): improves robustness

2. Searching for New Physics: Anomaly Detection

The problem: How do we find **unknown particles** or **rare exotic processes** in the huge LHC dataset, without knowing what exactly to look for?

AI solution: Use **unsupervised learning** to identify unusual events — things that don't look like Standard Model physics.

Key methods:

- **Autoencoders**: trained to reconstruct "normal" jets; large reconstruction error = anomaly
- **CWoLa (Classification Without Labels)**: trains on mixed samples, no signal labels
- **CATHODE**: combines unsupervised learning with generative models for better background estimation

CMS Case Study (2023):

- CMS trained unsupervised models on real data
- Found collisions with weird jet structures
- No new particle yet, but **proved that AI can search model-independently**

Why it's important:

- Extends physics reach **beyond human imagination**
- Doesn't rely on specific hypotheses
- Ideal for real-time anomaly hunting

3. Particle Tracking: GNNs for real-time reconstruction

The problem: Each collision creates **tens of thousands of particles** — tracking them all through detector layers is a massive combinatoric challenge.

AI solution: Use **graph neural networks** to build tracks from hit patterns.

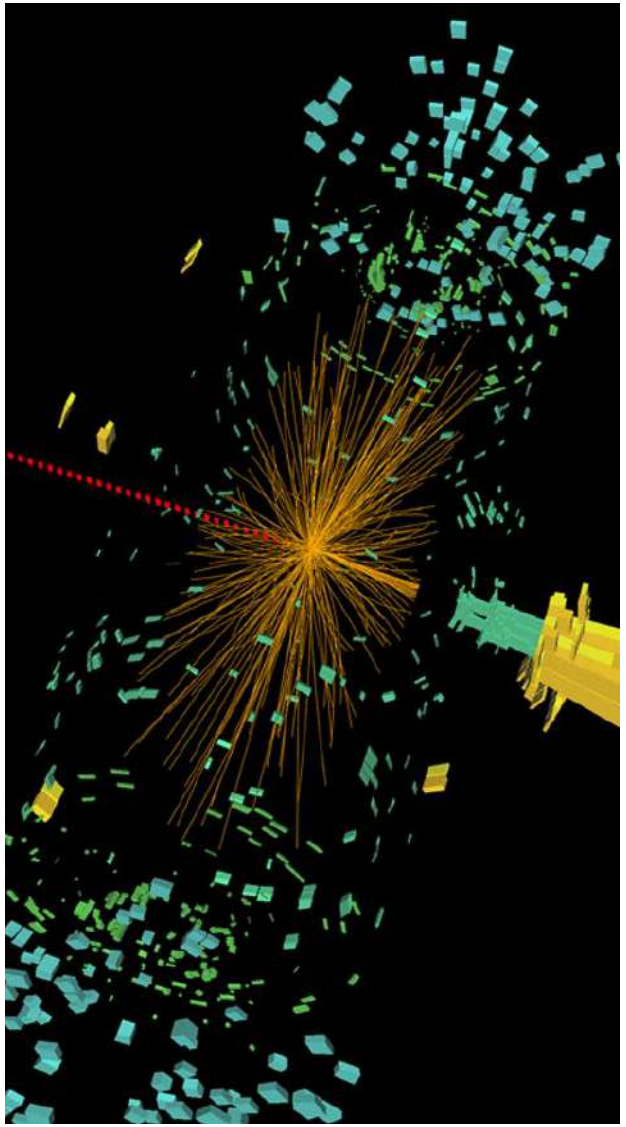
- Each detector hit = node
- Possible connections = edges
- GNN learns to recognize **valid trajectories**

Applications:

- LHC trigger-level tracking (latency < 4 μs)
- **TrackML challenge**: GNNs matched classical algorithms with much higher speed

Why it matters:

- Enables real-time physics decisions (keep or discard events)
- Will be **critical at HL-LHC**, with even higher data rates



4. Fast Simulation: Generative Models

The problem: Monte Carlo simulation (e.g., GEANT4) is **extremely accurate** but also **extremely slow** — up to minutes per event.

🧠 **AI solution:** Train a generative model to imitate simulation output in milliseconds.

Models:

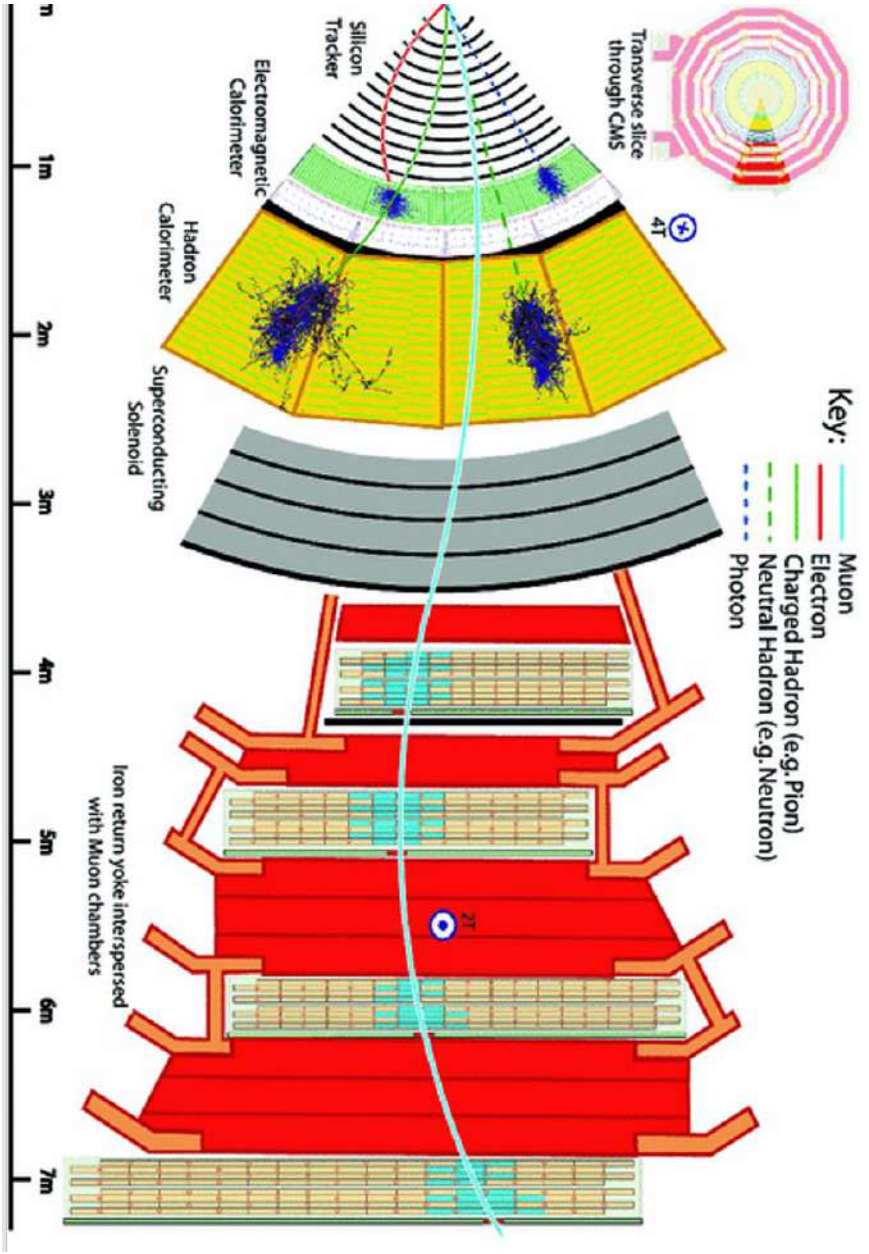
- **GANs / VAEs**: fast, but sometimes unstable or low-resolution
- **Diffusion Models**: new frontier; generate detailed detector responses with high fidelity

Examples:

- **CaloGAN**: simulates electromagnetic showers in calorimeters
- **DraPES (2024)**: diffusion model to generate signal+background events
- **Conditioned generation**: input particle type, energy, angle → get simulated detector output

Why it matters:

- Speeds up simulation by **factors of 1K-1M**
- Essential for testing theories, building training datasets, estimating backgrounds.



Applications in Astrophysics and Cosmology

Goal: Demonstrate how AI methods help astronomers extract discoveries from vast and complex datasets: sky surveys, gravitational waves, neutrinos, and exoplanets.

1. Real-Time Sky Monitoring: LSST and Transient Classification

The problem: The Rubin Observatory's **Legacy Survey of Space and Time (LSST)** will detect **~10 million transient events every night**: supernovae, flares, variable stars, or unknown phenomena.

AI solution:

- Use **machine learning models** to automatically:
  - Classify known event types
  - Detect **anomalies** — potentially *new* phenomena

Key points:

- **No fixed rule set** can cover all possibilities — AI generalizes better
- Alert streams are filtered in real time using classifiers
- Unknown events are flagged for human review ("The Unimagined")

Why it matters:

- AI becomes an **autonomous discovery engine**
- Filters out false positives, prioritizes real science

Camera 3.2B pixels

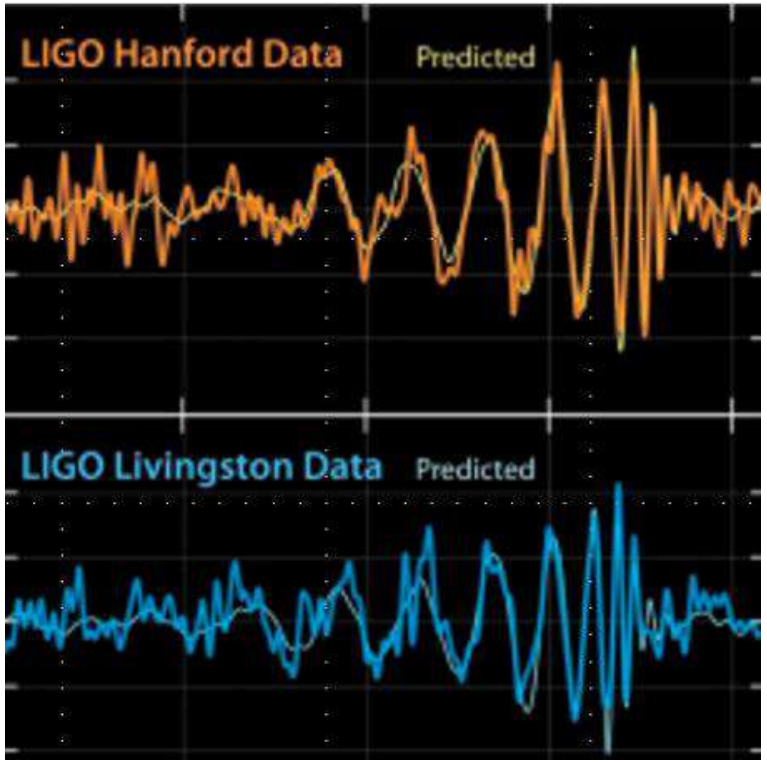
First photo: A mosaic of 1,185 frames taken over 10.5 hours over 7 nights - tens of millions of objects!

2. Gravitational Wave Detection: LIGO

The problem: LIGO detects tiny ripples in spacetime caused by **collisions of black holes and neutron stars** — buried in noisy data.

AI solution:

- **CNN ensembles** process long time-series data to identify merger signals
- Replaces **template-matching** (which is accurate but slow)



Achievements:

- In one test, AI scanned **a full month of LIGO data in ~50 seconds**
- **Noise filtering networks** trained to remove non-physical disturbances

Why it matters:

- Enables **real-time alerts** to point telescopes at fresh events
- Improves detector **sensitivity** by suppressing instrumental noise

3. Neutrino Detection: IceCube at the South Pole

The problem: High-energy neutrinos from space interact in Antarctic ice → create Cherenkov light → detected by 5000+ sensors. But:

- Geometry is complex
- Environment is inhomogeneous
- Signal-to-noise is low

AI solution:

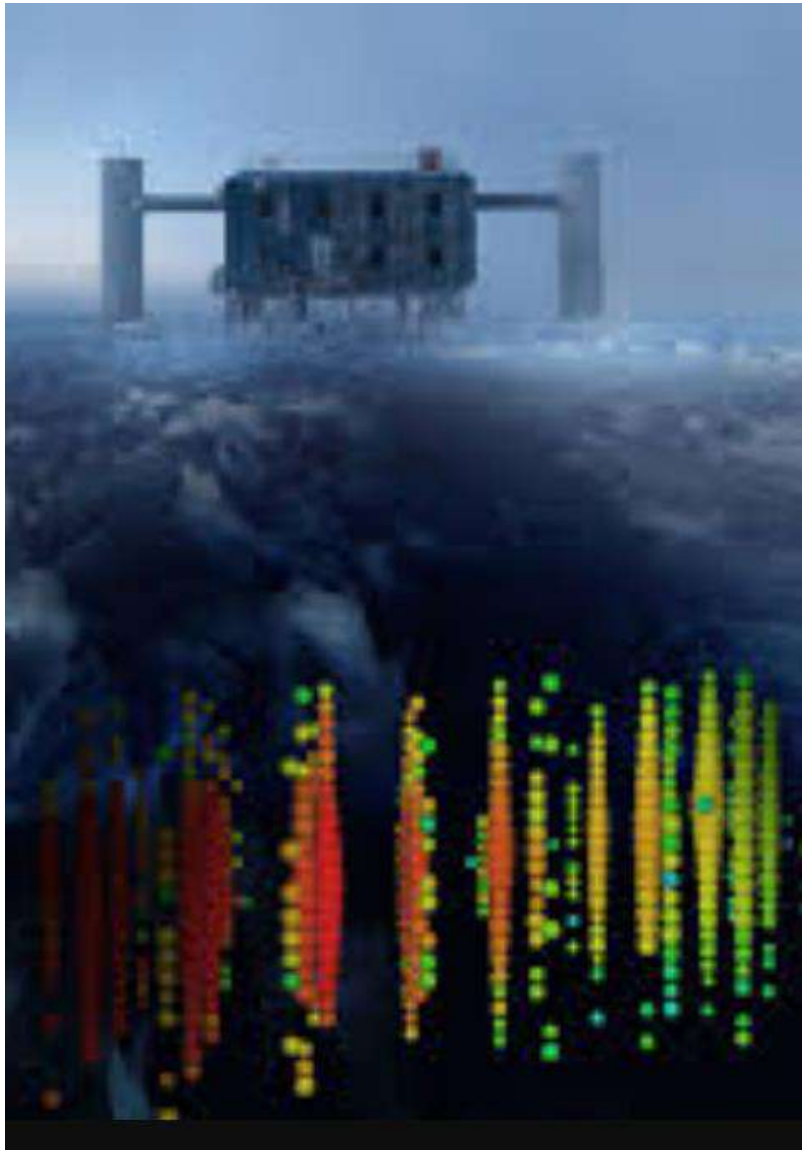
- Use **Graph Neural Networks (GNNs)**:
  - Treat sensors as graph nodes
  - Photon hits are signals
  - Predict **event type**, energy, direction

Result:

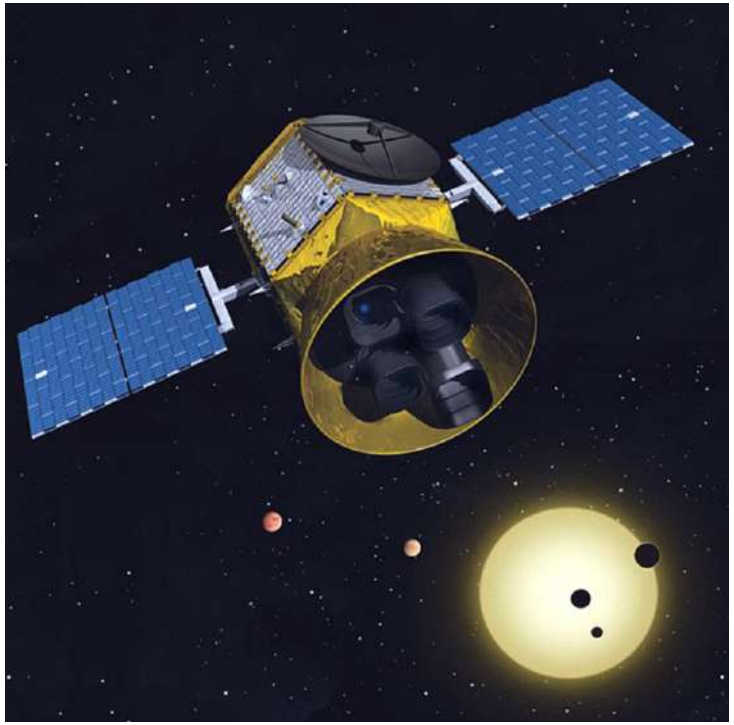
- The **GraphNeT model** reconstructs neutrino events **10,000× faster** than classical methods
- Enables near-instant alerts to the astronomical community

Why it matters:

- Improves sensitivity to cosmic sources (e.g., supernovae, black hole jets)
- Moves toward **real-time neutrino astronomy**



🌍 4. Exoplanet Detection: Kepler and TESS



The problem: Space telescopes like **Kepler** and **TESS** collect light curves from thousands of stars.
Planet transits = tiny brightness dips = hard to detect.

🧠 **AI solution:**

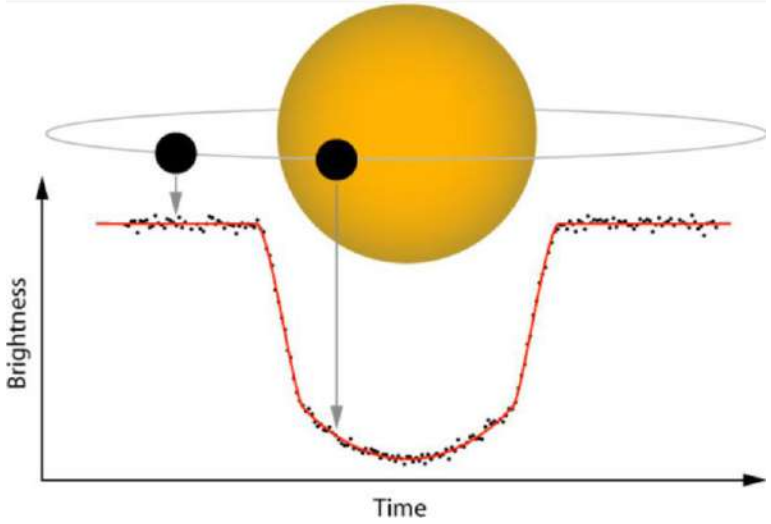
- **Deep neural networks** trained to spot transit patterns:
  - **ExoMiner**: validates planet candidates automatically
  - **AstroNet**: initial classification of light curves

Results:

- In 2021, **ExoMiner confirmed 301 new exoplanets**
- Networks are faster and more consistent than manual vetting

Bonus:

- Some models are **explainable**: they highlight features that influenced the decision (improves trust)



5. Some other Applications (very briefly in 1 line)

- **Galaxy morphology classification (spiral, elliptical) — with CNNs**
- **Photometric redshift prediction — with DL regressors**
- **Detection of gravitational lenses or merging galaxies — with anomaly detection**
- **Multimodal learning — combining spectra, images, time series**

Trends and Future Prospects

Goal: Summarize key emerging directions, open problems, and long-term potential of AI in physics — not just as a tool, but as a paradigm.

🌟 1. Explainable AI (XAI) and Human-Guided Models

The problem: Many AI models are **black boxes** — high-performing but hard to interpret.

Why it's critical in physics:

- We care **not only about the answer**, but *why* it's correct.
- If AI finds a signal — we need to trust and **physically explain** it.

Solutions:

- **XAI methods**: feature importance, saliency maps, surrogate models
- **Human-Guided AI (HG-AI)**: incorporate expert knowledge into training
  - Examples: adversarial de-biasing, physics-informed losses

⚠ 2. Systematic Bias and Data Shifts

⚠ 2. Systematic Bias and Data Shifts

The problem: ML models can pick up **unwanted correlations** — e.g., between jet class and jet mass — and produce biased results.

Examples:

- "Mass sculpting" in jet classification
- Misinterpreting detector artifacts as signals

Trends:

- Decorrelated training (mass-aware losses)
- Adversarial robustness
- Cross-checks across datasets

3. Reproducibility and FAIR Principles

Issue: AI models are often hard to reproduce across labs or experiments.

Solution:

- Apply **FAIR** principles (Findable, Accessible, Interoperable, Reusable) not only to data — but to **models themselves**.

Trends:

- Shared model repositories (e.g. HuggingFace for HEP)
- Open training pipelines and notebooks
- Citation-ready AI components in physics papers

4. AI in Real-Time Systems

Context: Almost all new experiments (e.g. LHC, LSST, LIGO, IceCube) generate **vast data flow**, which must be analyzed in real time process data (no way to store all info) **as it arrives** — filtering millions of events per second.

Use cases:

- **LHC trigger systems**: decide which collisions to keep
- **LSST alerts**: classify sky events within seconds
- **LIGO detection pipelines**: issue warnings for follow-up

Trend:

AI models optimized for **latency**, deployed on FPGAs/GPUs with ms-level response.

5. Specialized Models and Scientific Foundation Models

Direction:

- Instead of general-purpose AI, we're moving toward **narrow, domain-specific models**:
  - **Cosmosage**: LLM for astrophysics
  - **HEPsage** (future): physics-aware scientific assistant
  - Diffusion-based simulators trained on collider data
- Combine text, code, math, and physical priors

Goal: AI as a **research collaborator**, not just a tool:

1) Teacher
2) Code verifivation & writing
3) Idea generation & testing

6. Beyond Classical AI: Hybrid & Quantum Integration

Next frontier:

- **Physics-informed neural networks (PINNs)** — enforce known physical laws during training
- **Hybrid modeling** — combine simulations + ML
- **Quantum machine learning (QML)** — for high-dimensional spaces and optimization

7. In-Vivo Neural Networks — Physics Meets Biology

What is it? A neural network implemented directly inside **a living biological system** — such as brain tissue or living cells.

Key Concepts:

- **"In-vivo"** means “inside a living organism”
- Uses **biological neurons** or hybrid bio-electronic circuits to process information
- Learns from real-time physical stimuli (light, sound, movement)

Applications:

- Brain-machine interfaces
- Adaptive sensing and signal processing
- Real-time learning directly in physical systems (e.g., robotic skin, neural implants)

Why it matters in physics/AI:

- Merges **neuroscience and machine learning**
- Pushes the boundary between hardware and wetware
- Inspires new architectures for **energy-efficient, adaptive computing**

Challenges:

- How to do interfaces?
- Long-term stability (survival of living neurons)

Advantages:

- Efficiency.

🧠 7. AI for Discovery: Toward Autonomous Science?

- Generative models now create realistic events → could one day generate **new hypotheses**
- Unsupervised anomaly detection may discover **unexpected physics**
- Future experiments may include **AI agents** for online planning, model testing, or result vetting

- AI is not replacing physicists — it's **augmenting** them.
- The physicist of the future won't be the one who can simulate fastest, but the one who understands how to **ask better questions** with the help of AI.

The AI Scientist: Autonomous Research Agent

What is it?

A fully autonomous system that performs the entire scientific workflow:

- Generates hypotheses
- Writes and runs code
- Analyzes results
- Writes research papers
- Submits to peer review

Who built it?

- Developed by researchers at UBC, Oxford, and Sakana AI
- Paper: *"The AI Scientist" (arXiv:2408.06292, 2024)*

How it works:

- Uses large language models (LLMs) + tool access
- Constructs and explores experiments as a tree (agentic search)
- Optimizes code, plots, text, and logic without human intervention

Achievements:

- Papers accepted to real ML workshops
- Cost: under $15 per paper
- Demonstrates a working loop of *autonomous science*

Why it matters:

- First true example of end-to-end AI-driven research
- Opens the door to AI as a **research collaborator**, not just a tool

The Samurai Path to Data Analysis with AI

For physics students entering the world of data and AI

🎯 **Philosophy: You don’t need to be a software engineer to work with AI.**

Instead, aim to become a *physicist who can speak data and collaborate with AI tools.*

Stage I: Mindset — Why should a physicist care about AI?

- **AI won't replace physicists — but it may replace those who don’t use it.**
- Modern research in high-energy physics, astrophysics, and cosmology relies heavily on data and algorithms.
- Your job isn’t just to process data — but to **ask smart questions** and let AI help explore the answers.

🛠 Stage II: Tools of the Trade

1. Google Colab / Jupyter Notebooks

- Free, open environments for writing and sharing Python code
- Great for working with real physics datasets and visualizing results
- Fully cloud-based — no setup needed

2. Python (not C++)

- Friendly syntax, massive libraries: NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch
- You don't need advanced programming skills — just enough to manipulate data, train a model, and plot results

3. GitHub

- A version-controlled, open notebook of your growth
- Use it to share code, track versions, contribute to others' projects
- Treat it as your research portfolio

4. Kaggle

- Not just competitions — a community, a school, and a data playground
- Datasets on astrophysics, particle tracking, signal classification
- Learn by replicating kernels, joining discussions, analyzing public notebooks

Stage III: Working With AI as a Partner

- Use **ChatGPT / Claude / Gemini / Grok** not just to answer questions — but to:
  - Draft Python code
  - Explain errors and concepts
  - Generate hypotheses and test ideas
  - Write summaries, reports, or LaTeX drafts
- Treat AI as your **research assistant**: available 24/7, good with math, never tired.

Stage IV: Your First Physics-AI Projects

Start small and visible:

- Reproduce a known result using open data (e.g. LHC open datasets)
- Classify galaxies from SDSS or LSST images
- Use VAE/GAN to simulate toy detector data
- Build a simple anomaly detector for cosmic-ray events

Host it on:

- GitHub + Google Colab (code + readme)
- Kaggle (notebook + explanation)
- Tilda / Notion / Behance (visual portfolio)

🚀 Stage V: Leveling Up

- Subscribe to HEP and ML newsletters / Telegram channels
- Follow competitions, papers (arXiv-sanity, Papers with Code)
- Join student AI clubs, hackathons, open-source physics ML communities
- Follow geeks on Youtube doing funny AI-things.
- Ask professors for small research problems with open-ended data

✅ Final Checklist: Are You Walking the Path?

Skill	Status
Knows how AI/ML works conceptually	✅
Can use Jupyter/Colab with Python	✅
Has one AI+ physics project online	✅
Uses GitHub to organize code	✅
Can ask AI for help with code & ideas	✅
Understands metrics, experiments, basic models	✅

Final Takeaways — AI & Data analysis in Physics

- **AI is not magic — it's math + data + compute.**
  Understand the principles, and it becomes your ally.
- **Data is now central to physics.**
  From LHC collisions to galaxy surveys — learning to *work with data* is a core skill.
- **Neural networks are universal tools.**
  CNNs, GNNs, Transformers — each solves different physics tasks, from jet tagging to exoplanet discovery.
- **Modern AI is fast, scalable, and increasingly explainable.**
  It’s not just for image classification — it’s entering simulation, anomaly detection, and real-time systems.
- **You don’t have to become a software engineer.**
  You need to become a *physicist who speaks data*, collaborates with AI, and builds smart pipelines.
- **The future is hybrid: Human intuition + AI power.**
  AI doesn’t replace curiosity — it amplifies it.

🌍 Real-World AI Technologies — Impact Beyond Physics

1. AlphaZero (DeepMind, 2017)

- Self-learning neural network mastered chess, shogi, and Go from scratch
- No human strategies or databases — pure reinforcement learning
- Defeated Stockfish despite evaluating far fewer positions per second
  - 📎 DeepMind blog
  - 📎 Chess.com article
  - 📎 arXiv paper

2. Deepfake Technology

- AI replaces faces or voices in photos/videos with high realism
- Used in cinema, virtual characters, advertising — with ethical implications
  - 📎 Example video – Dior J'adore ad

3. Neuralink (Elon Musk)

- Brain-computer interface using neural decoding
- AI translates brain activity into motion commands
  - 📎 Neuralink demonstration

-rmberddima(),
Featuree 5, oprsc))
nize swihmizer: step();
THANK U 4 ATTENTION!