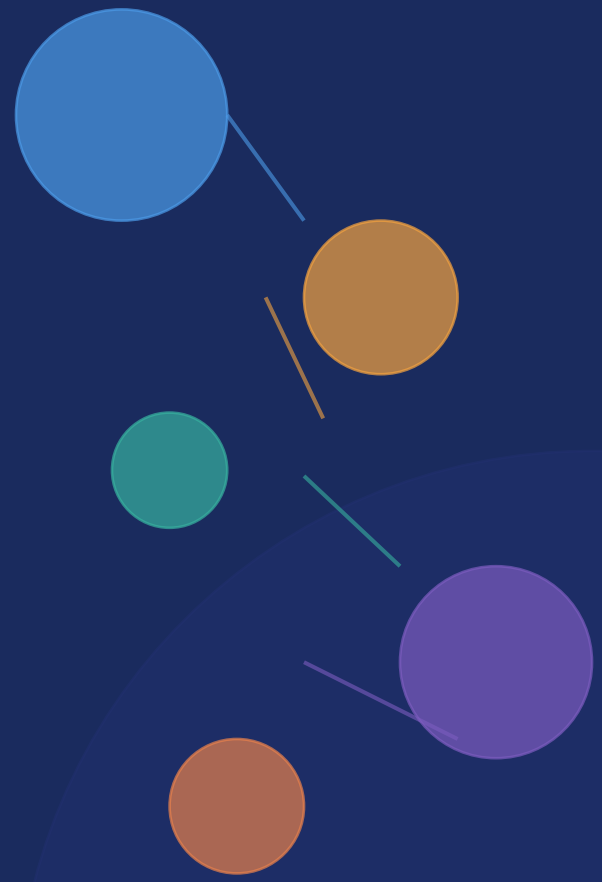
WEEK 01

The Genesis of Intelligence

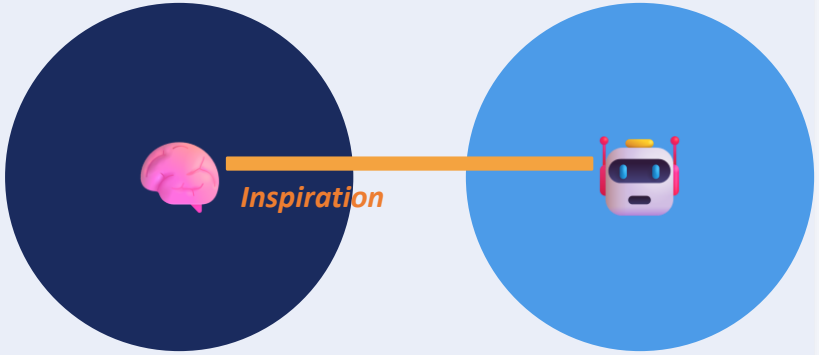
Core AI Principles & Agentic App Development

K. A. Ashan Priyadarshana | eLearning.lk



What is Artificial Intelligence?

"AI is the science of building machines that can perform tasks requiring human-like intelligence — reasoning, learning, perception, and decision-making."



Key Insight:

AI is not one thing — it's a family of techniques that evolved over 70+ years.

Reasoning
(තර්කනය)

Learning
(ඉගෙනීම)

Perception
(සංජානනය)

Decision-Making
(තීරණ ගැනීම)

Decision-Making: A combination of reasoning (understanding the problem), searching (exploring possible solutions in a solution space), and then deciding (choosing the best option based on goals and context).

Origins:

- Alan Turing proposed the 'Imitation Game' (1950)
- John McCarthy coined the term 'AI' (1956)
- Decades of research, data & compute led us here

The Genesis of AI — Founding Fathers & Key Milestones

"Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."

— Dartmouth Proposal, 1955

1950

Alan Turing

The Father of Computer Science

Wrote "Computing Machinery and Intelligence" — asked the bold question: "Can machines think?"

Proposed the Imitation Game (Turing Test): If a machine can fool a human into thinking it's human, it can be said to "think."

This paper lit the spark for everything that followed.

1956

John McCarthy

The Namer of AI

Organized the legendary Dartmouth Conference (1956) — the official "birth" of AI as a field.

Coined the term "Artificial Intelligence" and championed Symbolic AI — the idea that intelligence = manipulating symbols using logical rules.

Created LISP (1958), the language of AI research for decades.

1969

Marvin Minsky

The Bridge Builder

Co-founded MIT AI Lab. Built SNARC (1951) — one of the first neural network machines!

Ironically, later published "Perceptrons" (1969) with Papert, showing limitations of early neural nets — this shifted the field towards Symbolic AI for decades.

His "Society of Mind" theory: intelligence emerges from many simple, non-intelligent agents working together.

The Great Divide: Symbolic AI vs. Connectionism (Neural Networks)

Symbolic AI (McCarthy, Newell, Simon)
Intelligence = logical rules & symbol manipulation. "Neat" approach. Think like a philosopher — reason step by step.

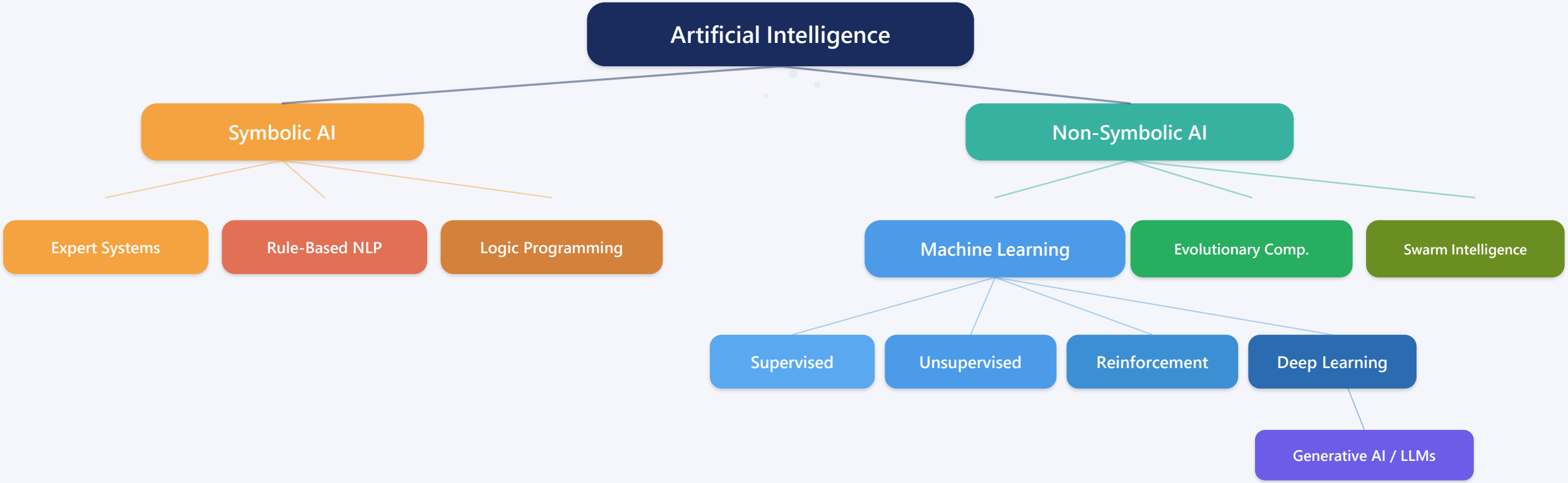
v s

Connectionism (Rosenblatt, later Hinton, LeCun)
Intelligence = networks of simple units learning from data. "Scruffy" approach. Think like the brain — learn from experience.

Key Milestones Along the Way

Fun Fact: The Dartmouth organizers believed 10 scientists working one summer could "solve" AI. 70+ years later, we're still working on it!

The AI Landscape — Your Map for Today



- **Evolutionary Computation** — No learning from data. Instead, solutions are **evolved** through selection, crossover, and mutation (e.g., Genetic Algorithms).
- **Swarm Intelligence** — Decentralized, self-organized systems inspired by collective behavior in nature (e.g., Ant Colony Optimization, Particle Swarm).

We will explore every node on this map today.

PART 01

Symbolic AI

When Intelligence Meant Rules

Core Concept:

Symbolic AI = Machines that manipulate human-readable symbols and explicit logic rules.
If you can write the rule down in plain language, the machine can follow it.

```
IF  temperature > 38°C  AND  cough = TRUE

THEN  diagnosis = "Flu"  →  prescribe treatment
```

← A real rule from the MYCIN expert system, 1970s

Expert Systems

Knowledge Base

A repository of encoded facts and domain-specific rules, built by interviewing human experts and transcribing their knowledge.

Inference Engine

Applies rules to inputs to reach conclusions using two strategies:

- **Forward Chaining** — data-driven; starts from known facts and fires rules until a conclusion is reached
- **Backward Chaining** — goal-driven; starts from a hypothesis and searches for supporting evidence

What problems do Expert Systems solve?

Medical Diagnosis

MYCIN (1970s) diagnosed blood infections — outperforming junior doctors at 65% accuracy

Legal & Tax Advisory

Systems that apply tax codes or legal statutes to case facts to recommend rulings

Equipment Fault Detection

Industrial systems that diagnose machine failures based on sensor readings

Loan Approval

Early bank systems that applied credit rules to approve or reject applications

Rule-Based NLP & Logic Programming

Traditional / Rule-Based NLP

Language processing using handcrafted grammar rules, dictionaries, and pattern matching — no learning from data.

Examples:

- Spell-checkers & grammar tools (MS Word, 1980s)
- Early spam filters (keyword matching)
- ELIZA chatbot — pattern-matched human responses (1966)
- Regex-based date/entity extraction

Key insight: You write the rules once. They work for everything that matches.

Logic Programming

Express knowledge as logical facts and rules. The engine uses deduction to answer queries — like a mathematical proof engine.

Examples:

- Prolog — parent(tom, bob). ancestor(X,Z) :- parent(X,Z).
- Theorem provers & math verification systems
- Legal reasoning & contract analysis
- Family tree & relationship inference queries

Key insight: Everything must be formally stated. Perfect precision, zero tolerance for ambiguity.

Symbolic AI — Strengths & Limitations

✓ Strengths

Fully Explainable

Every decision traces back to an explicit rule — you can always ask 'why?'

No Training Data Needed

Rules are hand-coded; you don't need thousands of examples to get started

Precise & Consistent

Same input always produces the same output — no randomness or drift

Works in Narrow Domains

Excellent for well-defined problems with clear, stable rules (e.g. tax code)

✗ Limitations

Breaks on Ambiguity

Natural language is messy. "Bank" means river bank or finance? Rules struggle.

Doesn't Scale

The real world has millions of exceptions. You can't write a rule for all of them.

Requires Expert Engineers

Someone must interview experts and manually encode every piece of knowledge (This is called Knowledge Engineering and it is hard)

Brittle to Change

When the domain changes (new laws, new diseases), all rules must be manually updated

"The world has too many exceptions for rules to handle alone. We needed machines that could learn."

THE SHIFT

From Rules to Learning

01

Data Exploded

The internet generated more data than humans could ever write rules for. Images, text, clicks — billions per day.

02

Rules Hit a Wall

Every exception needed another rule. Every edge case broke the system. The complexity was unmanageable.

03

Biology Offered a Blueprint

The human brain doesn't run on rules — it learns from experience. Researchers asked: can machines do the same?

Enter: Non-Symbolic AI → machines that learn patterns from data

PART 02

Non-Symbolic AI

When Intelligence Became Learned

Core Concept:

Non-Symbolic AI = No explicit rules. The machine discovers patterns in data and builds its own internal numerical representation — weights, not words.

Symbolic Approach

```
Rule: IF x THEN y
(written by a human expert)
```

Non-Symbolic Approach

```
Model: learns f(x) → y
from 1,000,000 examples
```

Machine Learning — Three Fundamental Paradigms

All machine learning approaches answer one core question: How does the machine receive feedback during learning?

🧑‍🏫 Supervised Learning

Learning WITH a teacher

Given labeled training examples (input + correct answer), learn to predict the right answer for new inputs.

📖 Analogy: A child learning from a picture book

Example problems:

Spam detection, house price prediction, disease diagnosis

🔍 Unsupervised Learning

Learning WITHOUT a teacher

Given data with no labels, discover hidden structure, groupings, or patterns on your own.

🗺 Analogy: Exploring a new city with no map

Example problems:

Customer segmentation, anomaly detection, topic modeling

🎮 Reinforcement Learning

Learning by DOING

An agent interacts with an environment, receiving rewards for good actions and penalties for bad ones.

🚲 Analogy: Learning to ride a bike

Example problems:

Game AI (AlphaGo), robotics, recommendation engines

Supervised Learning — In Depth

Two main problem types in Supervised Learning:

Classification — Discrete Output

"Which category does this belong to?"

- 📧 Email: Spam or Not Spam?
- 🏥 Medical: Is this tumor malignant or benign?
- 🖼️ Image: Cat, Dog, or Car?
- 💳 Finance: Is this transaction fraudulent?

Regression — Continuous Output

"What is the predicted value?"

- 🏠 What will this house sell for?
- 🌡️ What will tomorrow's temperature be?
- 📈 What will this stock price be in a week?
- ⏱️ How long will this delivery take?

Common Algorithms: Linear Regression · Logistic Regression · Decision Trees · Random Forests · SVM · Neural Networks

Unsupervised Learning — In Depth

No labels. No "right answer". The algorithm must discover hidden structure on its own.

Clustering

Group similar data points together without being told what the groups are.

Example problems:

- Customer segmentation by purchase behavior
- Grouping news articles by topic
- Detecting communities in social networks

K-Means · DBSCAN · Hierarchical Clustering

Dimensionality Reduction

Compress high-dimensional data to fewer dimensions while preserving key structure.

Example problems:

- Visualise 1000-feature datasets in 2D
- Remove noise from data before modelling
- Face recognition feature compression

PCA · t-SNE · Autoencoders

⚠ Anomaly Detection

Identify rare data points that differ significantly from the majority — no labels needed.

Example problems:

- Flagging unusual credit card transactions
- Detecting server intrusions
- Finding defective products on a line

Isolation Forest · One-Class SVM · Autoencoders

Reinforcement Learning — In Depth

An agent learns by trial and error — no labeled data, just a reward signal.

Game-Playing AI

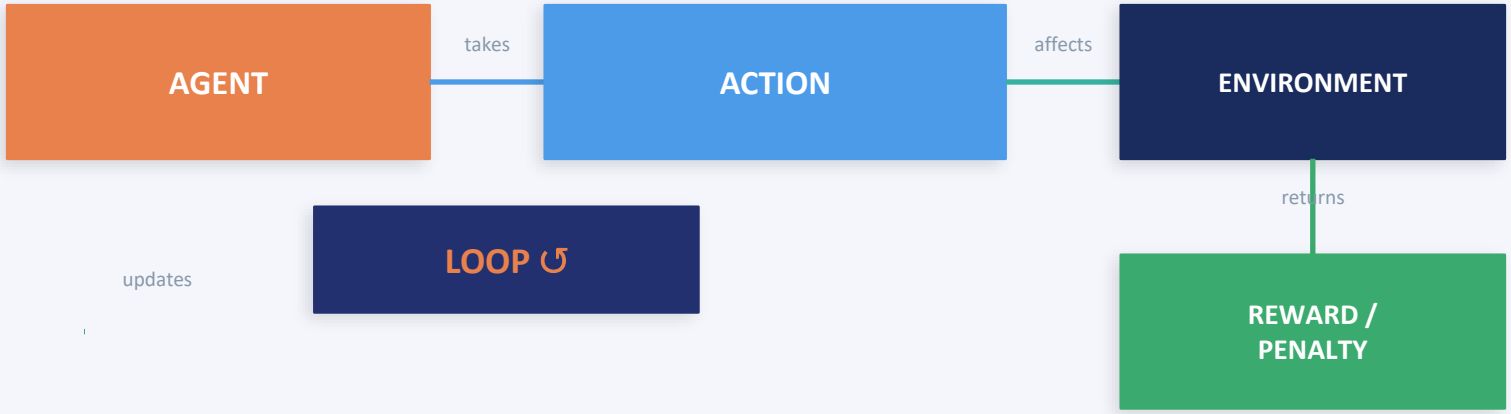
AlphaGo defeated the world Go champion. AlphaZero mastered Chess and Shogi with zero human game knowledge — just the rules and a reward for winning.

Robotics

Teaching a robot arm to pick and place objects. It attempts movements, fails, receives penalties, and gradually learns dexterous manipulation.

Recommendation Engines

YouTube and Netflix optimise watch time. Each click is a reward signal that shapes future recommendations.



Key insight: The reward signal IS the teacher. No labeled dataset needed — only a well-designed reward function.

Deep Learning — Going Deeper

A subfield of Machine Learning using neural networks with many layers.
Each layer learns increasingly abstract features from the data.

Why Deep Learning NOW?

More Data

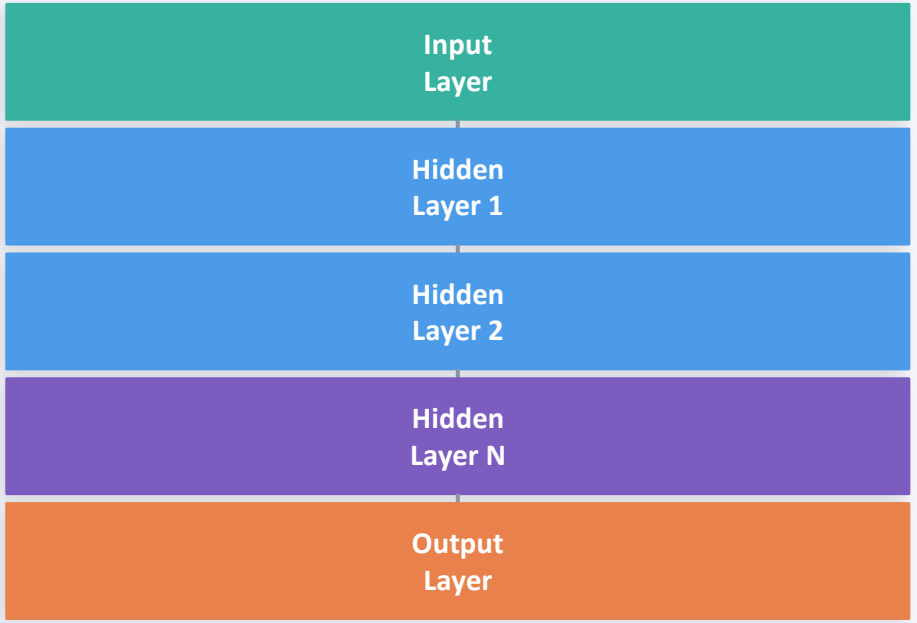
Internet-scale datasets (ImageNet, Wikipedia) gave models enough examples to learn from

Faster GPUs

Graphics cards enabled parallel matrix operations — training went from years to hours

Better Algorithms

Dropout, batch normalisation, and better activation functions solved key training problems



Example — Image Recognition:

› Layer 1 detects edges & lines

› Layer 2 sees shapes & corners

› Layer 3 recognises faces & objects

Three Architectures That Changed Everything

Each architecture was designed for a specific type of data structure:

CNN

Convolutional Neural Network

Sees: Patterns in SPACE (grids, images)

Slides small filters across an image to detect local features — edges, textures, shapes — then combines them hierarchically.

Use cases: Image classification · Object detection · Medical imaging · Self-driving cars

★ ResNet (2015) enabled 100+ layer networks; revolutionised computer vision.

RNN/LSTM

Recurrent Neural Network / Long Short-Term Memory

Sees: Patterns in TIME (sequences)

Processes inputs one step at a time, maintaining a 'memory' of previous steps — allowing context to carry forward.

Use cases: Speech recognition · Time-series forecasting · Language translation (pre-Transformer)

★ LSTMs solved the 'vanishing gradient' problem that made deep RNNs untrainable.

Transformer

Transformer (Attention Mechanism)

Sees: Patterns in CONTEXT (relationships between all tokens at once)

Uses self-attention to compare every word to every other word simultaneously — capturing long-range dependencies with no recurrence.

Use cases: GPT / Claude / Gemini · BERT · Code generation · Translation · This course (Weeks 4–8!)

★ 'Attention Is All You Need' (2017) — the paper that changed everything.

Generative AI — AI That Creates

Traditional AI: classifies, predicts, or decides. Generative AI: creates original content by learning the underlying patterns of training data.

Text Generation

- Long-form articles, stories, essays
- Code generation (GitHub Copilot)
- Conversational AI (GPT-4, Claude, Gemini)

Image Generation

- Photorealistic images from text prompts
- Art creation (DALL-E, Midjourney)
- Image editing and inpainting

Audio Generation

- Music composition (Suno, Udio)
- Voice cloning and synthesis
- Sound effect generation

Code Generation

- Complete functions from comments
- Bug fixing and refactoring
- Unit test generation

Important: Generative AI is built ON TOP of Deep Learning. It is the application layer, not a separate branch of the tree.

Large Language Models (LLMs)

The engine powering modern conversational AI — and the foundation of everything we build in Weeks 4–8.

What is an LLM?

A Transformer-based model trained on billions of words of text. Its core task is predicting the next token — but in learning to do this across the entire internet, it learns language, facts, reasoning, code, and much more.

~1T+	Trillions	Next Token
Parameters (GPT-4 est.)	Training Tokens	Core Training Objective

What do LLMs learn to do?

Conversation

Answer questions, explain concepts, assist with tasks

Reasoning

Solve multi-step problems, logical deduction, math

Coding

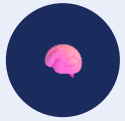
Write, debug, and explain code across languages

Writing

Summarise documents, draft content, translate languages

Tool Use

Call APIs, search the web, execute actions (agents!)



Knowledge

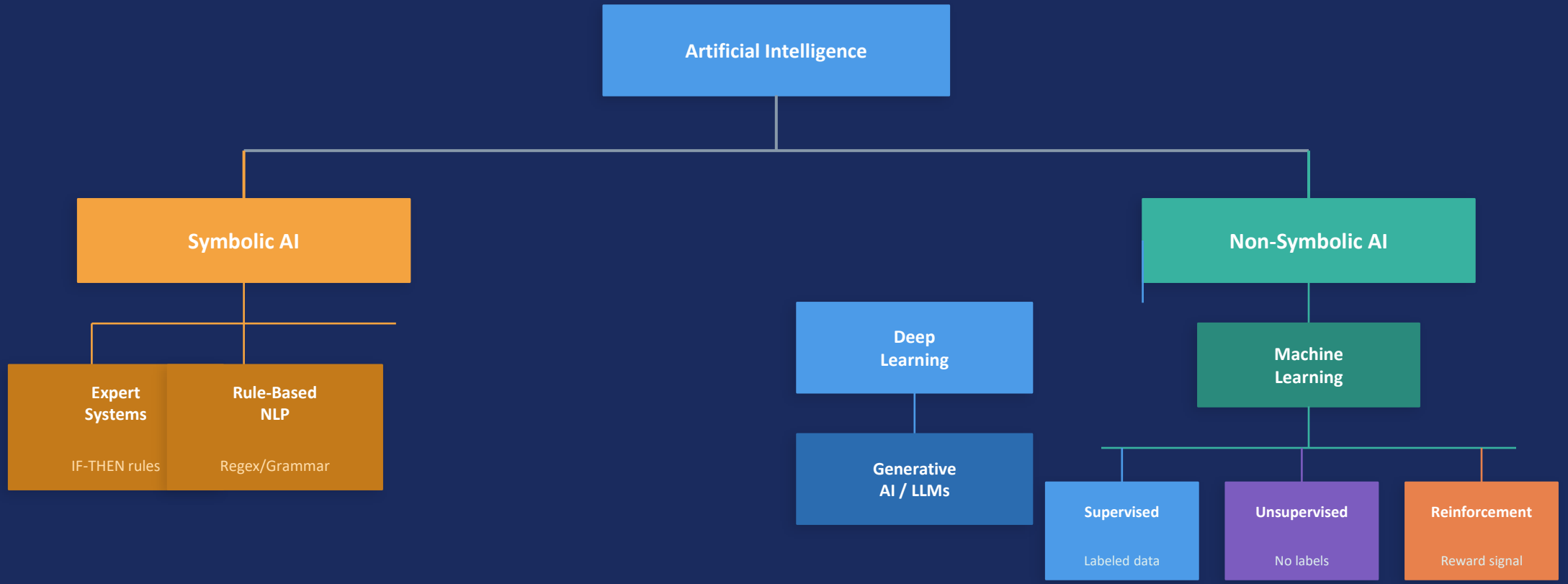
General world knowledge from training data

We will study LLMs deeply in Weeks 4 & 5 — and build agents on top of them in Weeks 6–8.

Your AI Algorithm Cheat Sheet

Algorithm	Problem Type	Example Problems	Core Idea
Expert Systems	Rule-based decisions	Medical diagnosis, tax advice	IF-THEN rules from domain experts
Rule-Based NLP	Text processing	Spam filters, early chatbots	Regex, grammar rules, dictionaries
Supervised Learning	Prediction / Classification	Email spam? House price? Disease?	Learn mapping from labeled input→output
Unsupervised Learning	Pattern discovery	Customer segments, anomaly detection	Find structure in unlabeled data
Reinforcement Learning	Sequential decisions	Game AI, robot control, recommendations	Maximize reward through trial & error
Deep Learning (CNN)	Vision tasks	Image classification, object detection	Hierarchical feature learning from grids
Deep Learning (Transformer)	Language tasks	Translation, Q&A, code generation	Self-attention over all tokens at once
Generative AI / LLMs	Content creation	Text, image, code, audio generation	Sample from learned data distribution

You now know every node on this map.



Legend

Symbolic AI | Non-Symbolic / ML | Deep Learning | Unsupervised | Reinforcement | Generative AI

Which Tool for Which Job?

Given a real problem, how do you pick the right AI approach?

If: Can you explicitly write out all the rules?

✓ **Use Symbolic AI**

Expert System or Rule-Based NLP

If: Do you have labeled training examples?

✓ **Use Supervised ML / DL**

Classification, Regression, or Transformer model

If: Do you have data but no labels?

✓ **Use Unsupervised Learning**

Clustering, Dimensionality Reduction, Anomaly Detection

If: Does an agent need to act & learn in an environment?

✓ **Use Reinforcement Learning**

Design a reward function; let the agent explore

If: Do you need to generate new content?

✓ **Use Generative AI / LLMs**

Fine-tune or prompt an existing model; or use an API

In real-world systems, multiple approaches are often combined. The best engineers know when to use each.

Week 1 Recap & What's Coming Next

Week 1 — Key Takeaways

- ✓ AI is a family of techniques, not a single thing
- ✓ Symbolic AI manipulates human-readable rules and symbols
- ✓ Expert Systems + Rule-Based NLP = key Symbolic AI tools
- ✓ Non-Symbolic AI learns patterns from data — no hand-coded rules
- ✓ ML has 3 paradigms: Supervised, Unsupervised, Reinforcement
- ✓ Deep Learning uses layered neural networks for complex patterns
- ✓ CNNs = images · RNNs = sequences · Transformers = language
- ✓ Generative AI creates new content by learning data distributions
- ✓ LLMs are Transformers trained on internet-scale text data

Week 2 — Coming Up

The Digital Neuron

Mastering the Learning Loop

Weights & Biases:

How a neuron processes and scales its inputs

Activation Functions:

The non-linearity that gives neural networks their power

Backpropagation:

How the network calculates its own errors and corrects itself

The Learning Loop:

Forward pass → Loss → Backprop → Update → Repeat

Questions?

Week 1 Complete — The Genesis of Intelligence

Core AI Principles & Agentic App Development

In Partnership with

eLearning.lk

Next Session: Week 2 — The Digital Neuron →