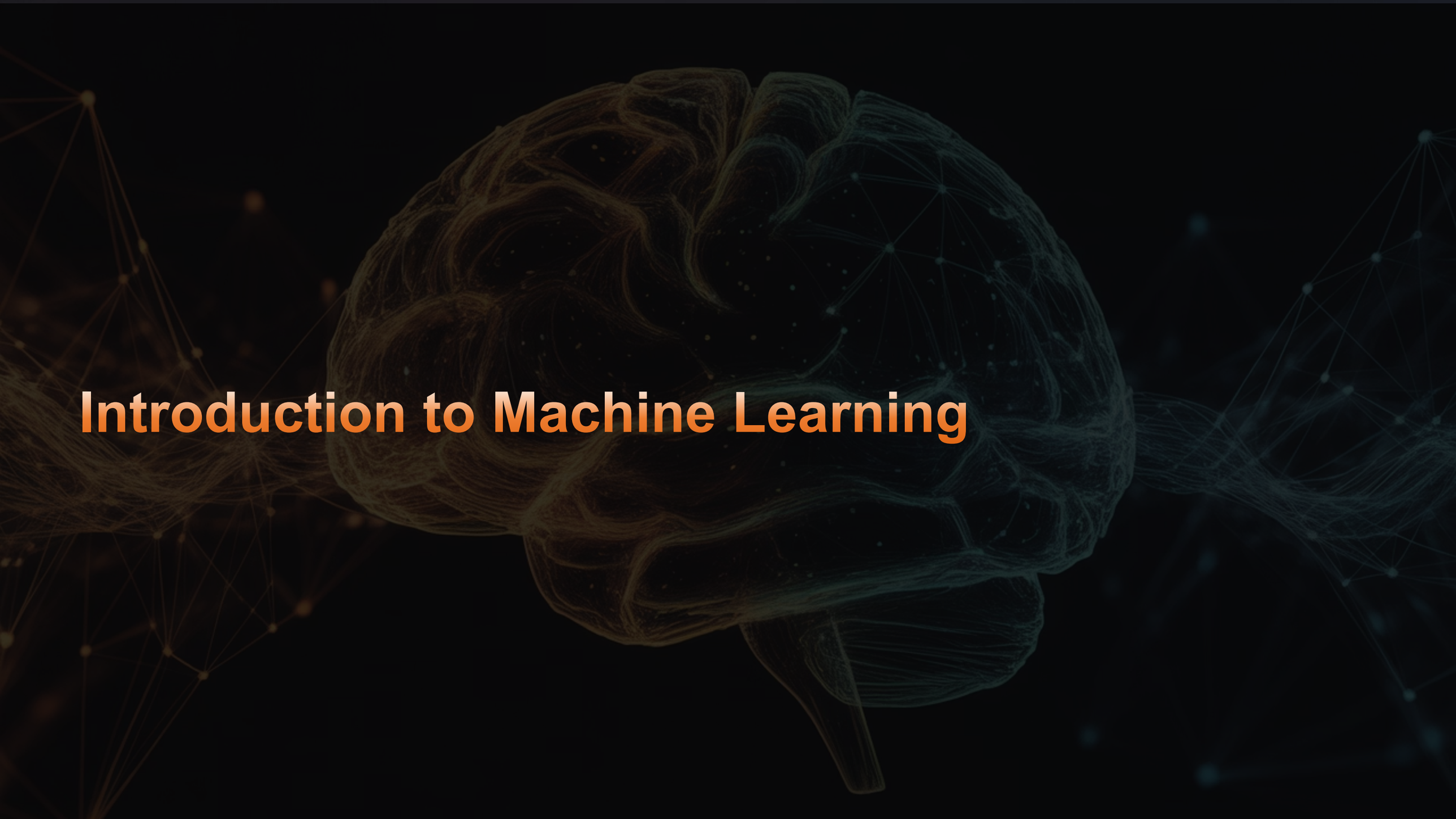

Introduction to Machine Learning

Early History & Progress of Machine Learning

- Since computers were invented, researchers have asked: Can machines learn?
- Machines still cannot learn as well as humans — but progress is significant.
- Effective algorithms now exist for specific learning tasks.

Potential uses:

- Medical diagnosis & treatment recommendations
- Smart homes optimizing energy use
- Personal assistants adapting to user interests

Early History & Progress of Machine Learning

Proven applications:

Speech recognition

Landmark developments by Waibel (1989) and Lee (1989).

Medical predictions

Such as pneumonia recovery rates (Cooper et al. 1997).

Credit card fraud detection

Utilizing patterns to identify suspicious transactions.

Self-driving vehicles

Early work like Pomerleau (1989) demonstrated capabilities on public roads.

Game playing

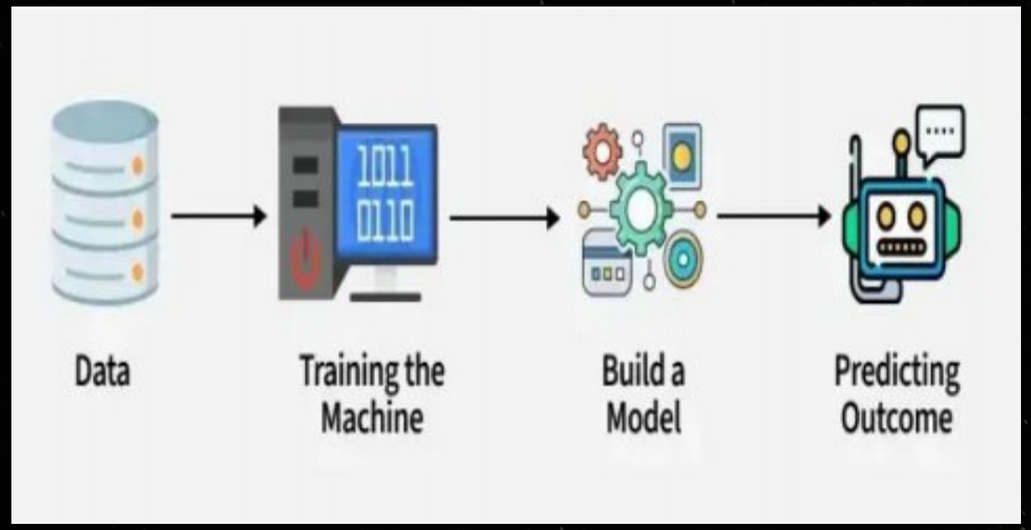
Achieving near world-champion level in backgammon (Tesauro 1992, 1995).

Machine learning outperforms older methods in tasks like speech recognition.

Widely used for data mining: extracting knowledge from large databases (maintenance logs, loans, finance, medical records).

What is Machine Learning?

Machine Learning means teaching computers to learn from examples (data) and make predictions.



Well Posed Learning Problem

A computer program is said to learn from experience E with respect to some class of **tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

Clearly define three things:

Task (T)

What the program should do.

Experience (E)

How it learns (data, examples, practice).

Performance Measure (P)

How you measure if it's getting better.

Well Posed Learning Problem

Example — Email Spam Filter

Task (T): Classify emails as spam or not spam.

Experience (E): The program learns from examples of emails labeled "spam" or "not spam."

Performance Measure (P): The percentage of emails correctly classified (accuracy).

Example: Robot Driving Learning Problem

Task: driving on public four-lane highways using vision sensors

Performance measure: average distance traveled before an error (as judged by human overseer)

Training experience: a sequence of images and steering commands recorded while observing a human driver

DESIGNING A LEARNING SYSTEM

Choosing the Training Experience

Where will the learner get its experience or data? Will it get clear feedback or figure things out on its own?

Choosing the Target Function

What exactly do we want the learner to predict or decide? This is the goal or "rule" we want it to discover.

Choosing a Representation for the Target Function

How will the learner store or express what it has learned? (e.g., decision trees, rules, neural networks, etc.)

Choosing a Function Approximation Algorithm

What learning method or algorithm will we use to find the target function from the training examples?

Step 1: Choosing the Training Experience

•What data will the system learn from?

•Ensure data is:

- **Relevant**
- **Sufficient**
- **Accurate & Clean**

Examples of training data sources

•Historical databases (banking customer data)

•Images (for medical diagnosis)

•Live sensor data (IoT devices)

Note: Poor data → poor learning (Garbage In → Garbage Out)

Step 2: Choosing the Target Function

The **target function** defines *what mapping* the ML system is trying to learn.

•We want a function **f(x) → y**

- **x** = input data
- **y** = output prediction

Example:

•Input: Student study hours

•Output: Predicted exam marks

Here, **f(hours) = marks**

Step 3: Choosing the Representation for the Target Function

The learning system must **represent knowledge** in some form.

Representation affects:

Accuracy

Interpretability

Computational efficiency

Representation	Used In	Example
Logical rules	Expert systems	if fever & cough → flu
Decision Trees	Classification	Tree for loan approval
Linear Models	Regression	Sales = a × TV ads + b
Neural Networks	Deep learning	Face recognition

Step 4: Choosing a Function Approximation Algorithm
Now we need an **algorithm** that approximates the target function from training data.
Algorithm selection depends on:

- Data size
- Data type (numeric, text, image)
- Speed vs accuracy requirements

Category	Algorithms	Use Case
Supervised Learning	Linear Regression, SVM, Decision Trees, Neural Networks	Prediction from labeled data
Unsupervised Learning	K-Means, PCA	Discover patterns without labels
Reinforcement Learning	Q-Learning	Learn through feedback (rewards)

Step 5: The Final System Design
Once trained, the system must be:

- **Validated** (testing on unseen data)
- **Optimized** (tune hyperparameters)
- **Deployed** (made available for real-world use)
- **Monitored** (performance tracked continuously)

Important: Models degrade over time due to changing conditions → retraining is necessary.

Choosing the Training Experience

When you design a learning system (like a checkers-playing program), how you train it really matters!

There are three big questions to think about:

1. Feedback
2. Who Controls the Training
3. Does the Training Look Like Real-World Testing?

Choosing the Training Experience

Understanding how feedback is given is crucial for effective AI training.

Direct Feedback:

The program receives clear, exact guidance, often with explicit instructions on what constitutes a 'best move' or correct action at each step.

Example: An expert human player identifies the **best move** for a checkers program in every board position.

Easy to learn — the program immediately knows right from wrong, accelerating the learning process.

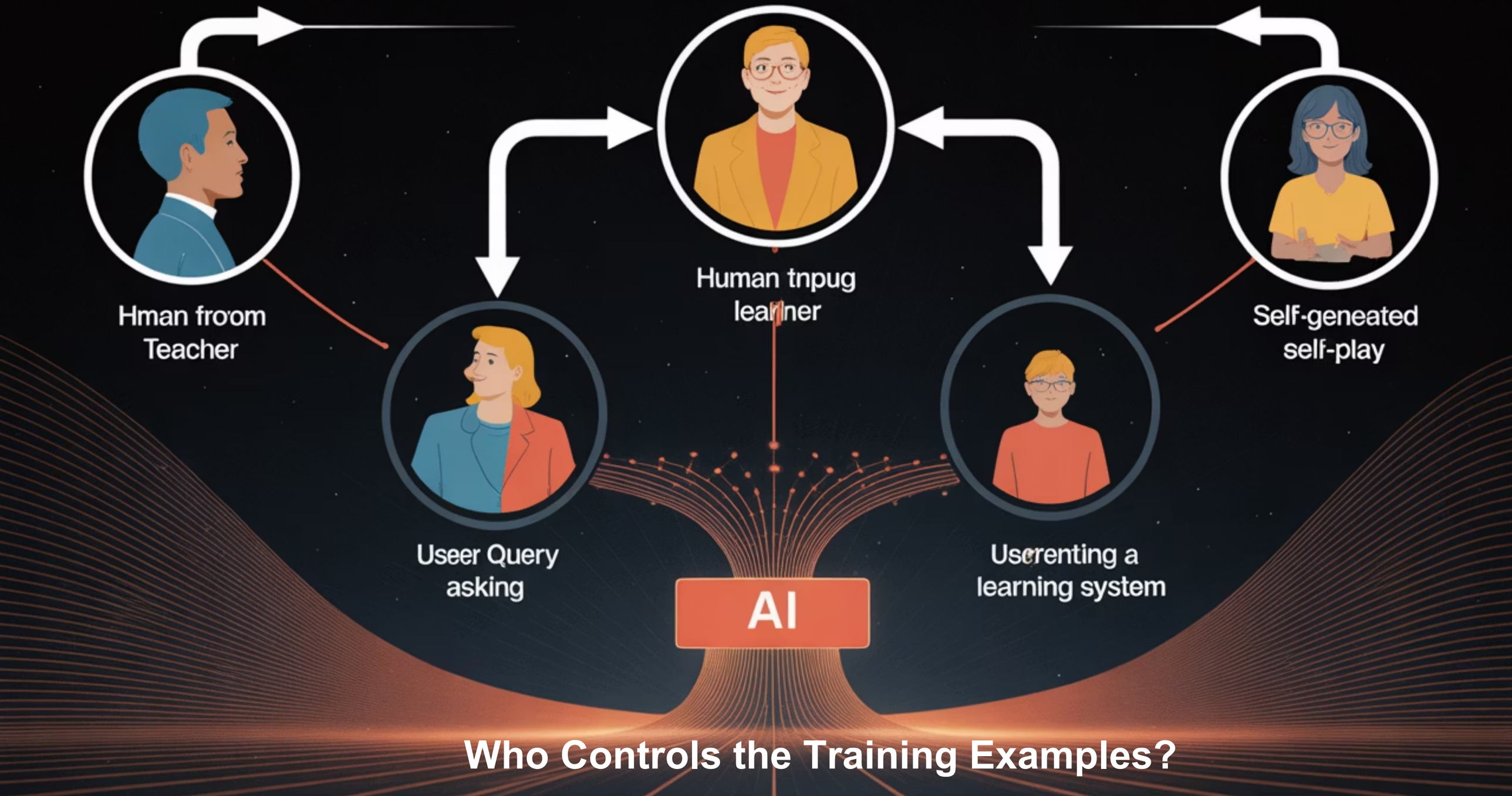
Indirect Feedback:

The program only gets a final outcome signal (e.g., win or loss) after completing a sequence of actions, without specific step-by-step correction.

Example: A checkers program learns by playing many games and only knows if it **won or lost** the entire game.

Harder — the program must infer which of its past moves were good or bad based on the ultimate result, a complex credit assignment problem.

Choosing the Training Experience



Who Controls the Training Examples?

Type of Control	Who Chooses the Training Examples?	Description	Example
Teacher-Controlled Training	External Teacher / Human / Environment	The learner (ML algorithm) is given labeled training examples selected by a teacher or supervisor.	In Supervised Learning, a teacher provides input-output pairs like (Email → Spam/Not spam).
Learner Requests Examples (Active Query / Active Learning)	Learner (the ML model) chooses which examples to learn from	The learner actively asks for specific examples that will reduce its uncertainty the most.	A model training a medical diagnosis system may request more examples of rare diseases to improve accuracy.
Learner-Controlled (Self-Play)	The learner generates its own training experiences	The learner learns by interacting with an environment, receiving rewards, and improving over time.	Reinforcement Learning, like AlphaGo, learns by playing millions of games against itself (self-play).

Choosing the Training Experience

Does the Training Look Like Real-World Testing?

If the training games look like the real games it will play later, it'll perform well.

If it only trains against itself, it might never see clever tricks that a real human champion uses.

So, training must be similar to the real test situations for the best results.

Choosing the right training experience means:

1. Give clear feedback if possible.
2. Make sure the learner sees examples similar to what it will see in the real world.
3. Decide who controls what it learns from — a teacher, the learner itself, or both.

Choosing the Target Function

When you design a learning system, you must decide: What exactly do you want the program to learn to do?

You want to build a computer program that predicts how much a house will sell for.

- **Task (T):** Predict house price
- **Experience (E):** Past data about houses — size, location, number of bedrooms, age, and the actual price they sold for.
- **Performance (P):** How close the predicted price is to the real selling price.

Direct Way: Predict the Exact Price

The obvious goal is:

Input: Details about the house

Output: Predicted price in dollars

So the **target function** is:

Price: H→R where,
H is the set of all possible houses (with features like size, location, etc.), and
R means it outputs a real number (the price).

Choosing the Target Function

But what if direct prediction is too complex?

Suppose the housing market is huge and prices depend on complex factors — neighborhood trends, school ratings, upcoming development projects — some of which are hard to model directly.

In that case, you might simplify the target function:

Instead of predicting the exact price

Predict a score: how desirable is the house?

$$Desirability \rightarrow [0, 100]$$

Choosing a Representation for the Target Function

The computer does not store knowledge the way humans do.

It needs a clear, formal way to store, modify, and use what it has learned.

The representation you choose decides:

1. How easy it is for the learner to learn it.
2. How well it works for real problems.
3. How fast predictions can be made.
4. Whether humans can understand it!

Choosing a Representation for the Target Function

Spam Email Classifier

Target function: Is this email spam or not? f:E→{spam,not spam}

Set of Rules

"IF email contains 'win lottery' AND has an unknown sender THEN mark as spam."

Decision Tree

A tree where you ask questions:

- Does the subject contain 'Congratulations'?
- Does the sender have a trusted domain?
- What is the probability of spam?

Linear Model

A mathematical formula like:

Spam Score=2.3×(Has 'free')+1.1×(Contains link)−0.5×(Sender know

Neural Network

A complex network inspired by the human brain, learning intricate patterns from data to classify emails effectively.

Choosing a Function Approximation Algorithm

1. **Decide what you want to learn (the target function)**
2. **Decide how you'll represent it (rules, decision tree, neural network, etc.)**

…the next big question is:

HOW will you actually ***learn*** it from the data?

This is where the **Function Approximation Algorithm** comes in.

A **Function Approximation Algorithm** is the *learning method* that adjusts your chosen representation to fit the training dat

Decision Tree

If your representation is a decision tree, you need a tree-learning algorithm to build it from data.

Neural Network

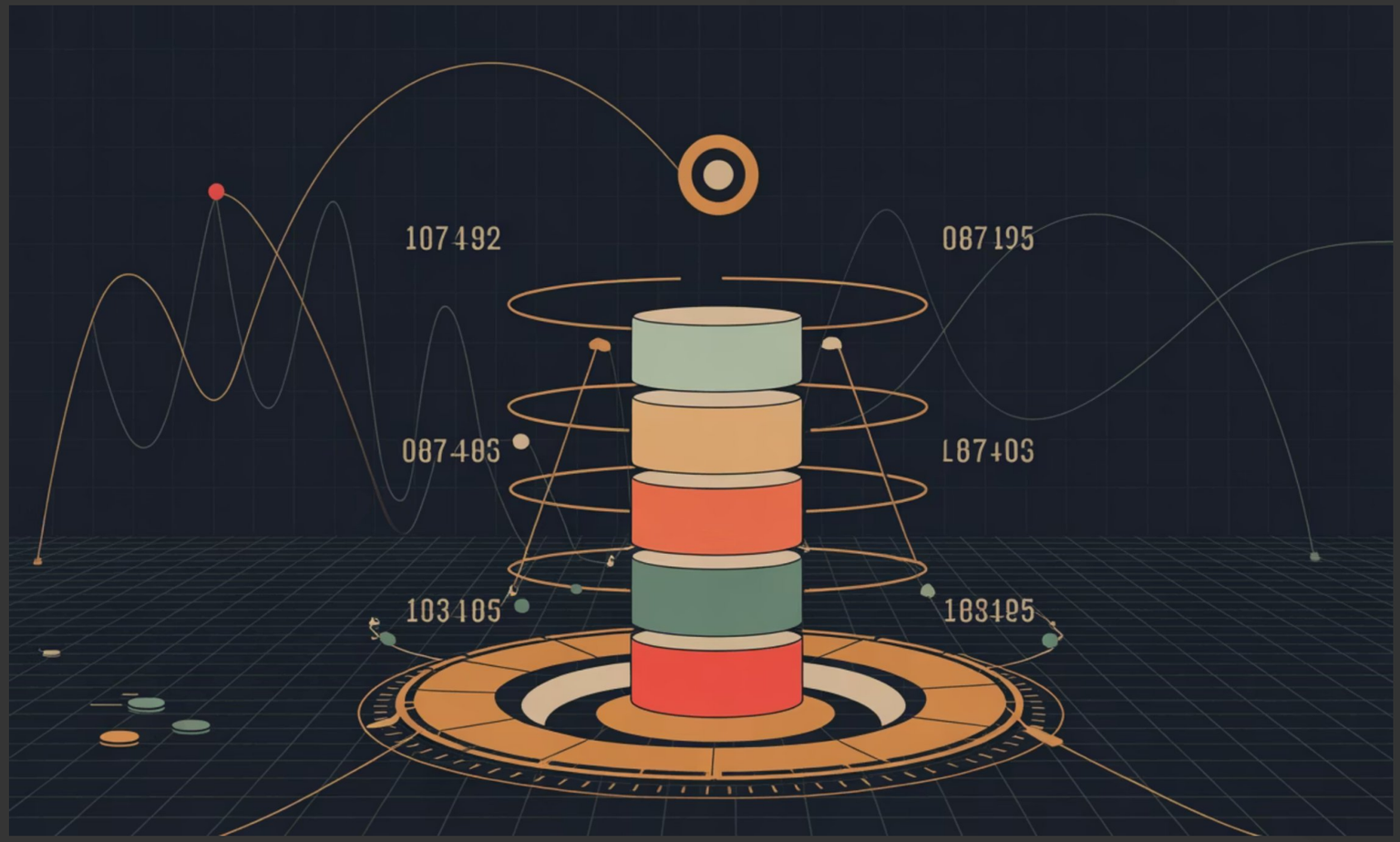
If your representation is a neural network, you need an algorithm that adjusts its weights to learn patterns.

Line or Curve

If it's a line or curve, you need an algorithm that finds the best line or curve to fit the given data points.

Go, change the

ADJUSTING THE WEIGHTS



For each training example, you know:

- The true value (from training data)
- The predicted value (from your model)

How Machine Learning Works

Collect Data

Gather examples (numbers, text, images, etc.)

Train Model

Show examples to the computer to learn patterns.

Test Model

Check learning using new examples.

Use Model

Apply it to real-life predictions.

Example: **Like teaching a child: show many apples and bananas, test with a new fruit.**

How Machine Learning Works

Let's say you're teaching a child to recognize fruits. You show 100 pictures — 50 apples and 50 bananas. For each picture, you tell the child the correct name.

That collection of pictures with their names is what we call data in machine learning.

In machine learning, a computer is like that child. It doesn't know anything by itself. We need to show it a lot of examples (data) so it can learn the difference.

Input Data

The examples you give (like pictures of fruits, sounds, or text).

Labels

The correct answers for each example (like "apple" or "banana").

Learning

The machine looks at the data again and again to find patterns.

Prediction

After learning, it tries to guess new examples by applying what it learned.

How Machine Learning Works

Overfitting:

A teacher teaches only from the last 5 years' question papers. Students do well if the same questions repeat — fail badly if even slightly new ones come.

Underfitting:

The teacher is vague, just talks theory with no examples or practice. Students don't learn how to answer any questions at all.

What is Train-Test Split?

We divide data into two parts:

- Training Set (80%): Used to teach the model.
- Test Set (20%): Used to check learning.

In machine learning, data is split into training data to learn patterns and testing data to evaluate performance.

For example, a utility company predicting future power demand may use 80% of smart meter load data for training and 20% for testing to ensure the model makes accurate real-world forecasts.

If we train and test on same data, model may memorize answers.

We want the model to understand patterns, not memorize.

Example: Reading answers before exam vs truly understanding the topic.

Example (Smart Grid Electricity Demand Forecasting)

A power utility company wants to build a machine learning model to predict the next day's electricity demand. They have 2 years of hourly load data collected from smart meters.

Step 1: Collect the Dataset

Total data = 17,520 records (2 years × 365 days × 24 hours)

Step 2: Split the Data

To train and evaluate the model, the data is divided into two parts:

Data Portion	Percentage	Purpose
Training Data	80% (≈ 14,000 records)	Used to teach the ML model patterns in electricity usage.
Testing Data	20% (≈ 3,520 records)	Used to check how well the model predicts unseen future load.

Step 3: Model Training

•The training data is used to develop relationships between:

- ◦Time of day
- ◦Weather condition
- ◦Historical load
- ◦Consumer behavior

The model learns patterns like:
“Load is higher in the evenings”
“Load increases on hotter days due to air conditioning”

Step 4: Model Testing

- The trained model is now evaluated on the testing data, which it has never seen before.
- If predictions are close to the actual values → model works well.

- If not → adjust model / features / algorithm.

Why Split the Data?

•To avoid overfitting (model only memorizing training data).
•To ensure the model can predict future unseen data accurately.

Evaluation Metrics (Judging the Model)

Accuracy

How many predictions were correct?

Precision

Out of predicted 'YES', how many were actually 'YES'?

Recall

Out of actual 'YES', how many were caught?

F1 Score

Balance between Precision & Recall

Prospects and Issues in Machine Learning

Prospects (Advantages)

- Automation of complex decision-making
- Improves efficiency and accuracy
- Personalization (e.g., ads, recommendations)
- Enables new innovations (autonomous vehicles, healthcare diagnostics)

Issues and Challenges

Issue	Explanation
Data Quality	ML is only as good as the data it learns from
Bias in Data	Can lead to unfair or discriminatory outcomes
Privacy Concerns	Sensitive user data must be protected
Interpretability	Complex models like Neural Networks are black-boxes
Computational Cost	Training large models requires high computing power

Conclusion

Machine Learning is a powerful technology enabling systems to learn and improve automatically.

Designing a good ML system requires thoughtful decisions in:

- Choosing meaningful data,
- Representing the learning task,
- Selecting suitable algorithms,
- And ensuring continuous evaluation.
- Despite challenges like data bias and privacy, the future of ML is promising and continues to revolutionize every industry.

Summary of ML Process

1

DATA

2

TRAINING

3

MODEL

4

TESTING

5

EVALUATION

6

DEPLOYMENT

THANK YOU

Go, change the world