Module 2 · Session 3

Neural networks, DL and Generative AI

The AI Spectrum — From Classical AI to Agentic AI

Leadership with AI – Rajendra M Sonar, IIT Bombay

What we are going to learn

Deep learning, artificial neural networks, NLP and foundations of Generative AI

1. Deep Learning
Limitations of machine learning, problem complexity and deep learning

2. Artificial neural networks
Foundations of neural networks, problems solved by neural networks, limitations and issues with neural networks

Deep Learning and Generative AI

3. NLP
Analyzing text data, NLP problem types, issues and limitations of NLP

4. Generative AI
Foundations of generative AI, contextual intelligence, prompt and context engineering, problems addressed by Generative AI, issues and limitations.

5. Integrating classical AI and Generative AI
How classical AI can be leveraged using Generative AI and how classical AI can address issues with Generative AI

Example: Problem complexities and artificial neural networks

Acquire → Learning

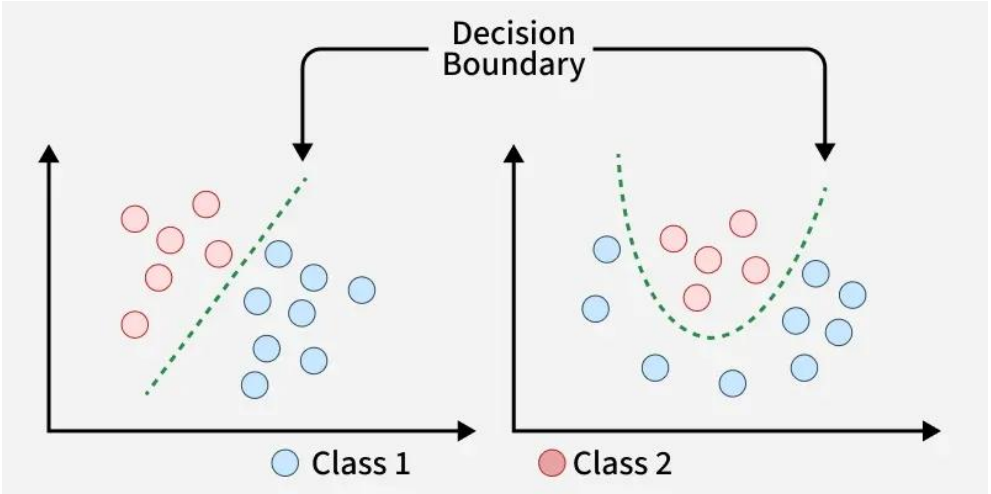
Memory
Adaptability

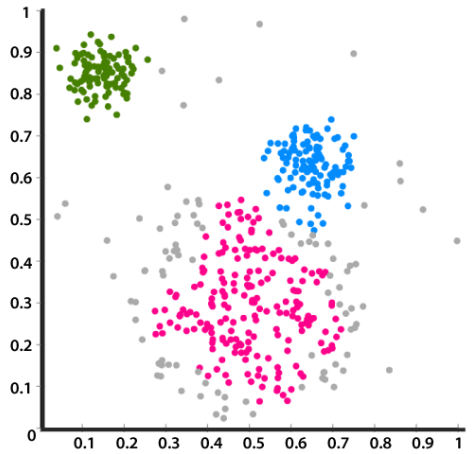
Tacit | Explicit | Implicit

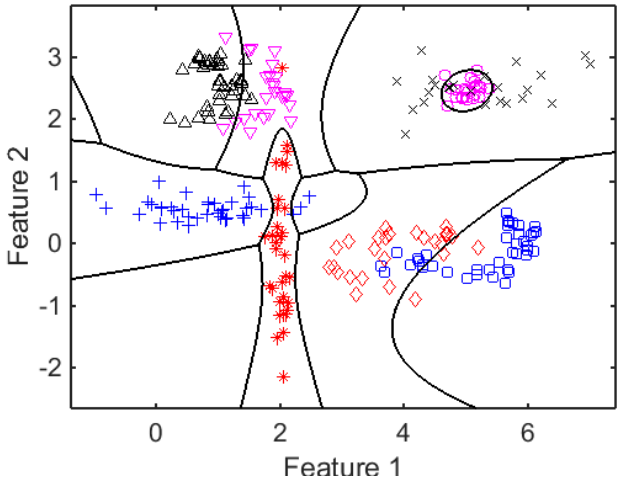
Apply

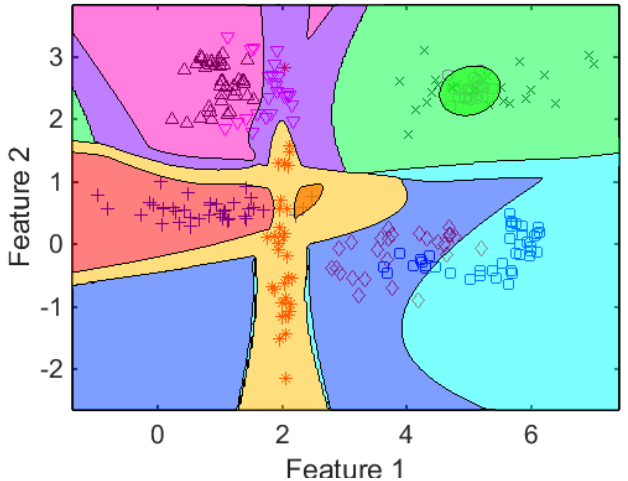
Think, plan & reason

Problem → Solution









Example: Problem complexities and artificial neural networks

Acquire → Learning

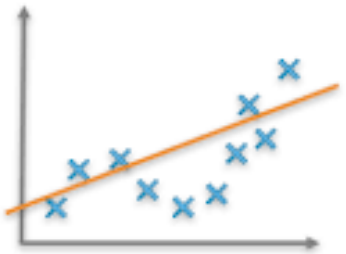
Memory
Adaptability

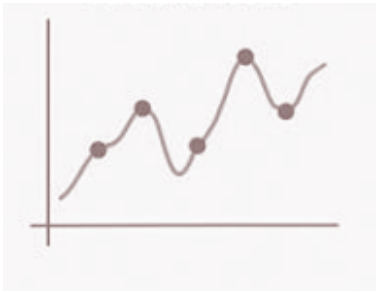
Tacit | Explicit | Implicit

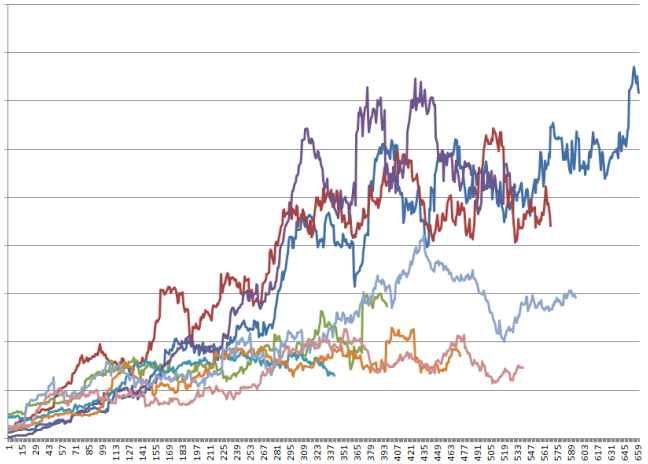
Apply

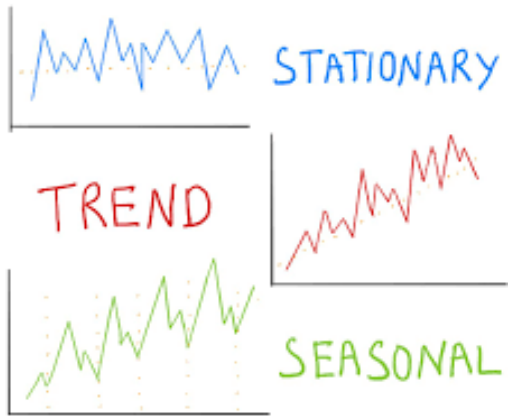
Think, plan & reason

Problem → Solution









Example: Problem complexities and artificial neural networks

Acquire → Learning

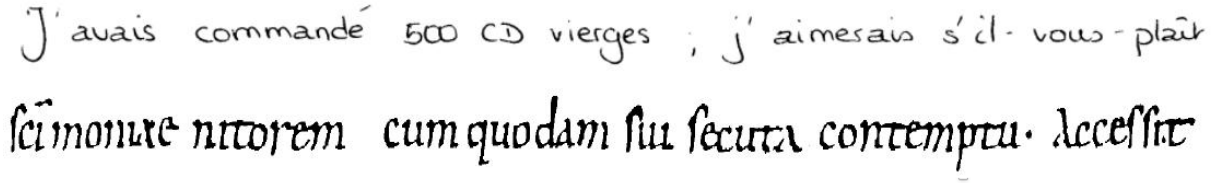
Memory
Adaptability

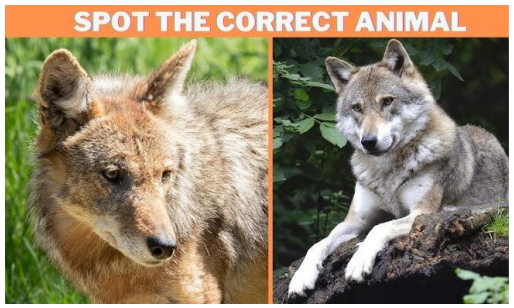
Tacit | Explicit | Implicit

Apply

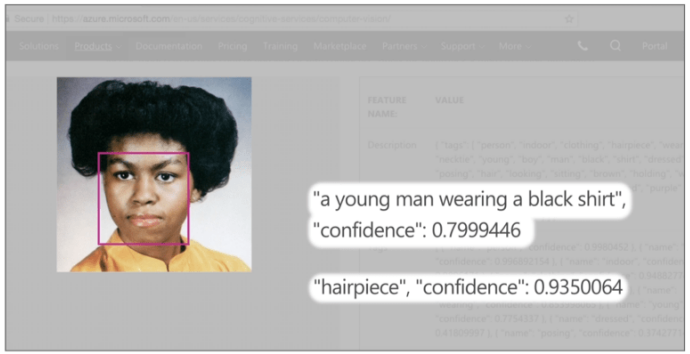
Think, plan & reason

Problem → Solution





Michelle Obama



Microsoft

Example: Problem complexities and artificial neural networks

Acquire → Learning

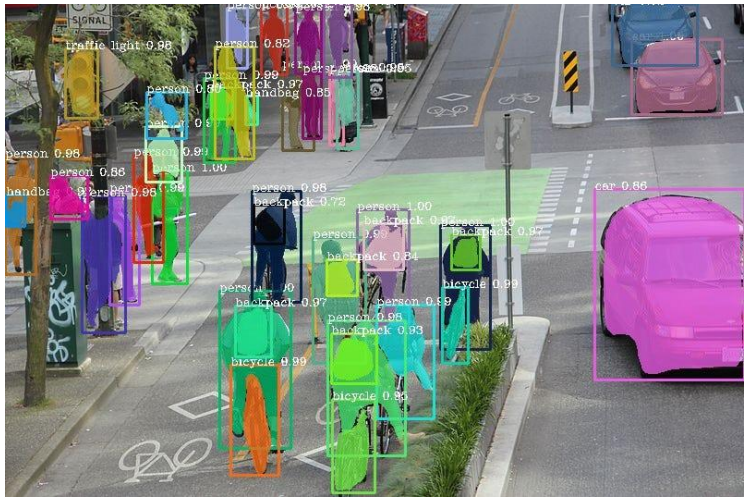
Memory
Adaptability

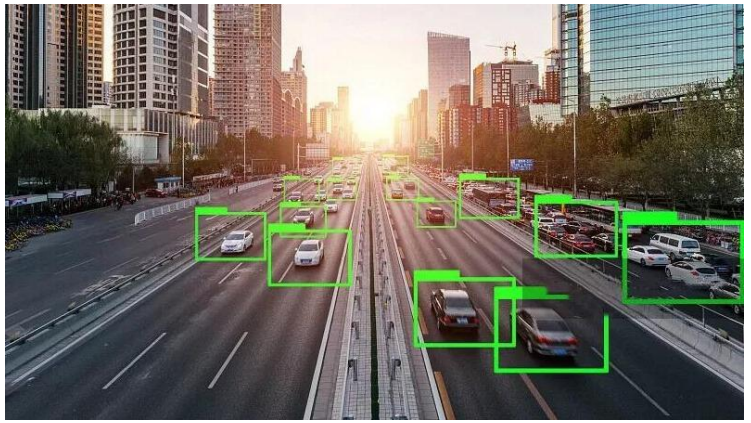
Tacit | Explicit | Implicit

Apply

Think, plan & reason

Problem → Solution





Example: Problem complexities and artificial neural networks

Acquire → Learning

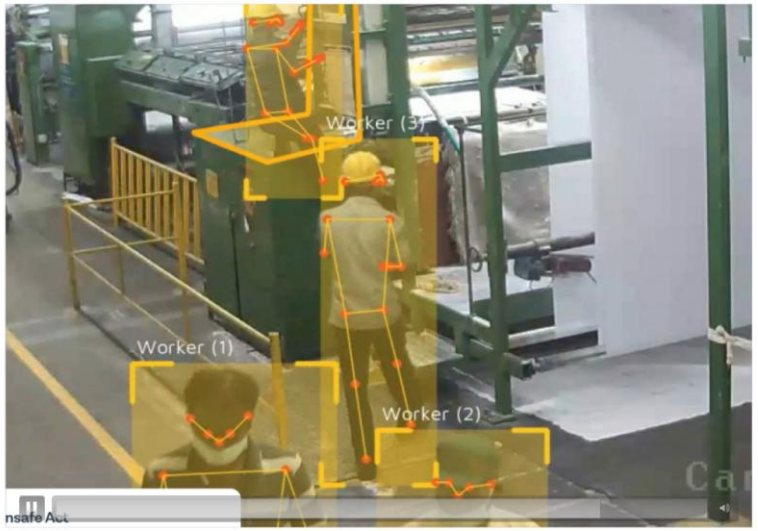
Memory
Adaptability

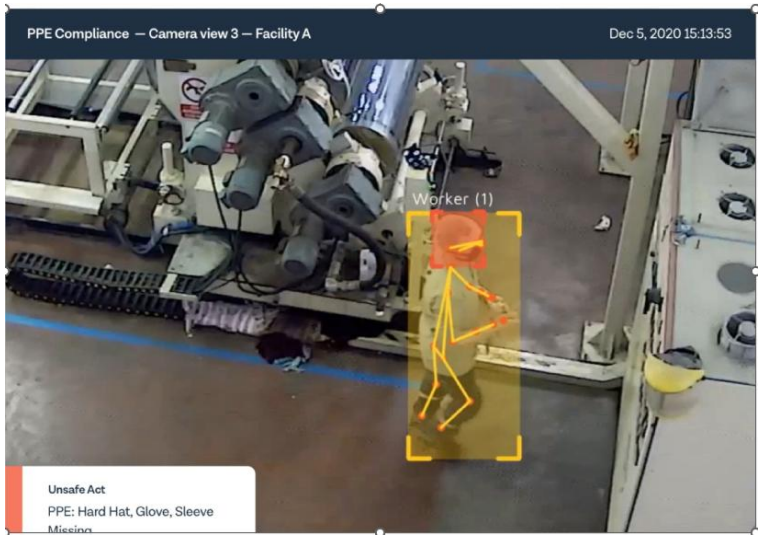
Tacit | Explicit | Implicit

Apply

Think, plan & reason

Problem → Solution





Intenseye: Powering Workplace Health and Safety with AI (A) | Harvard Business Impact Education

Example: Problem complexities and artificial neural networks

Acquire → Learning

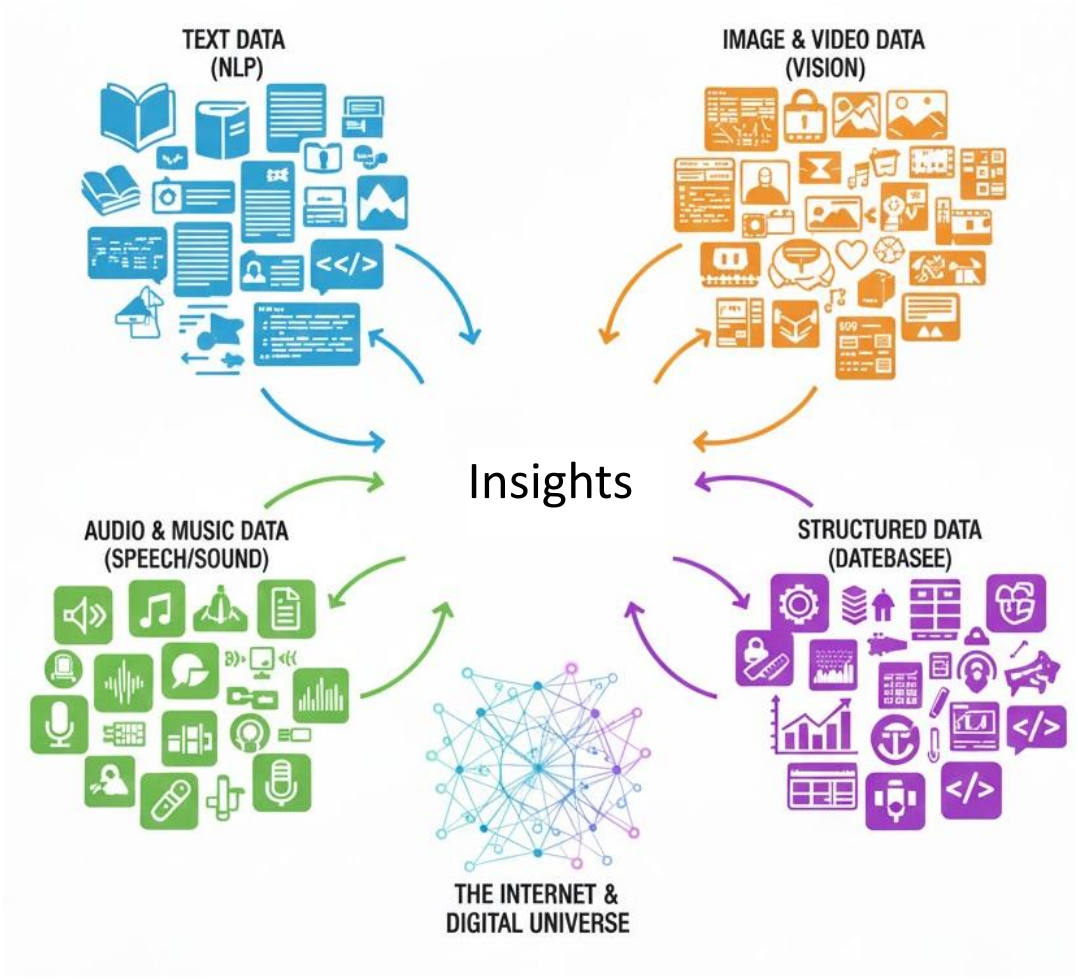
Memory
Adaptability

Tacit | Explicit | Implicit

Apply

Think, plan & reason

Problem → Solution



Example: Artificial Neural Networks: Artificial Neuron

Acquire → Learning

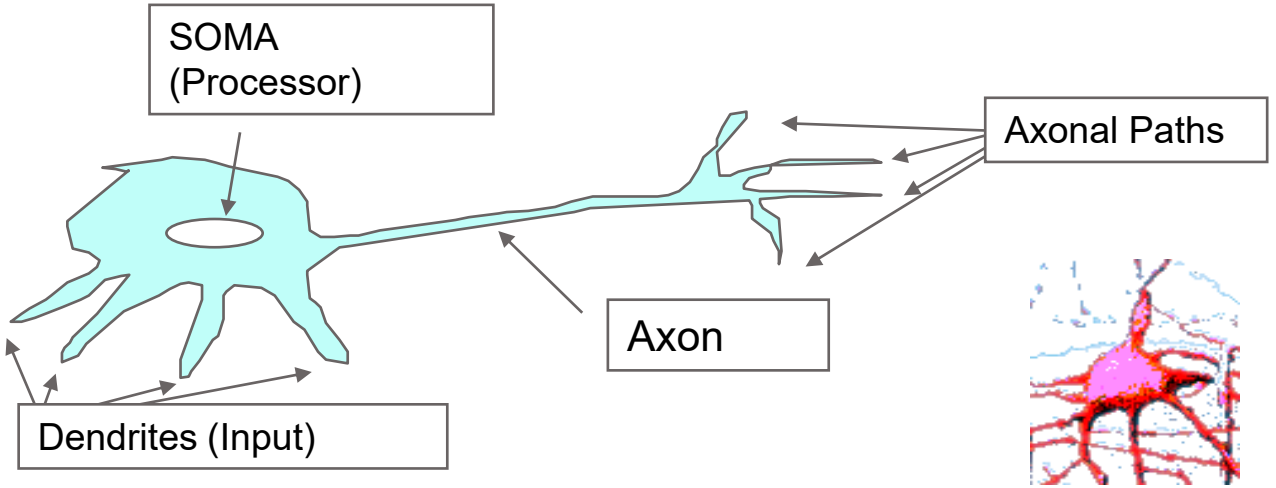
Memory
Adaptability

Tacit | Explicit | Implicit

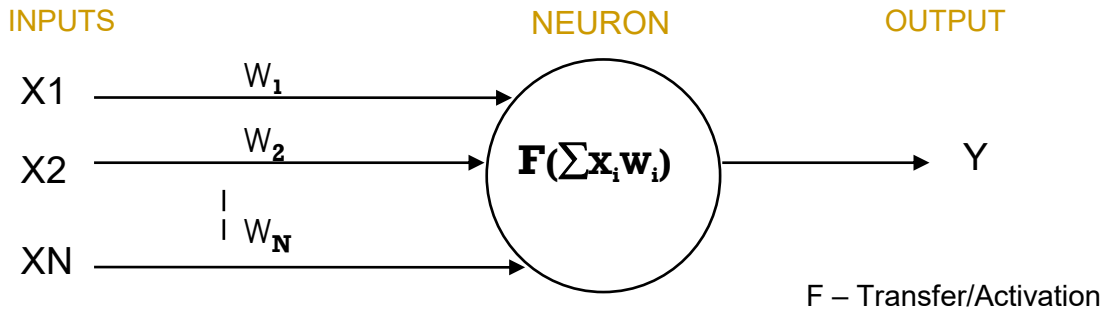
Apply

Think, plan & reason

Problem → Solution



Artificial Neuron: Simulation using mathematical model



F – Transfer/Activation function

Example: Artificial Neural Networks: Artificial Neuron

Acquire Learning

Memory
Adaptability

Tacit | Explicit | Implicit

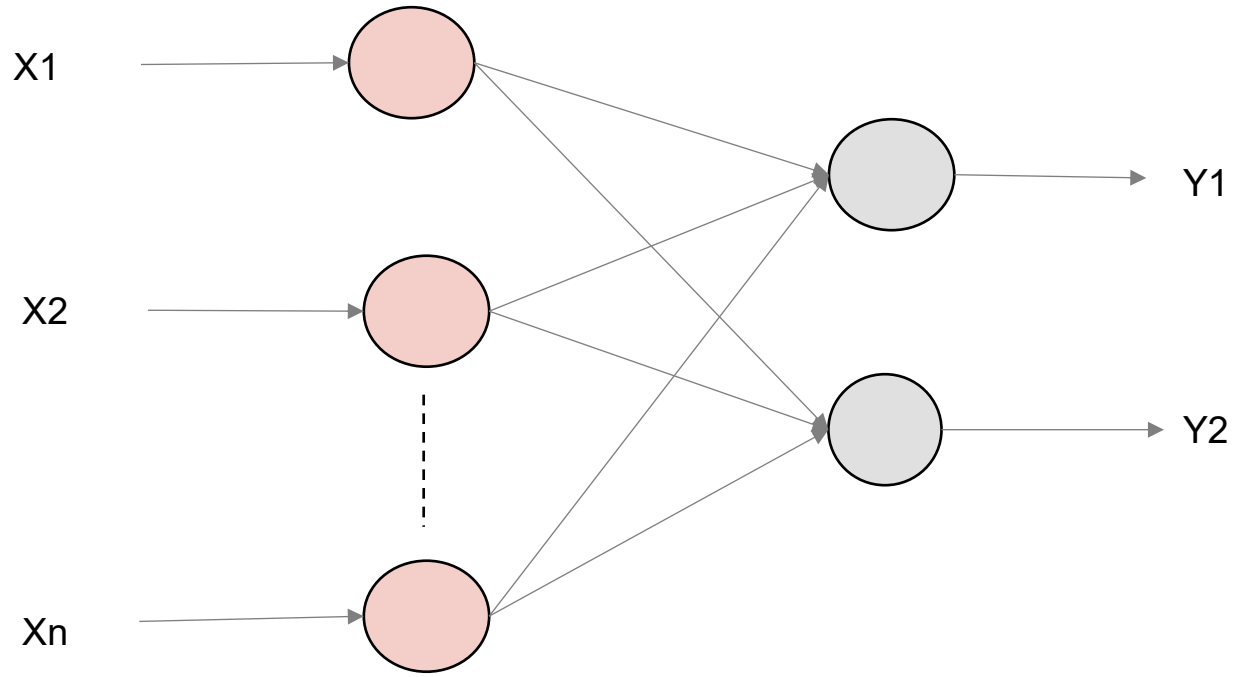
Apply

Think, plan & reason

Problem

Solution

Two Layer ANN Yi=f(Xi)



Inputs	Input Layer	Output Layer	Output/s

Example: Artificial Neural Network

Acquire Learning

Memory
Adaptability

Tacit | Explicit | Implicit

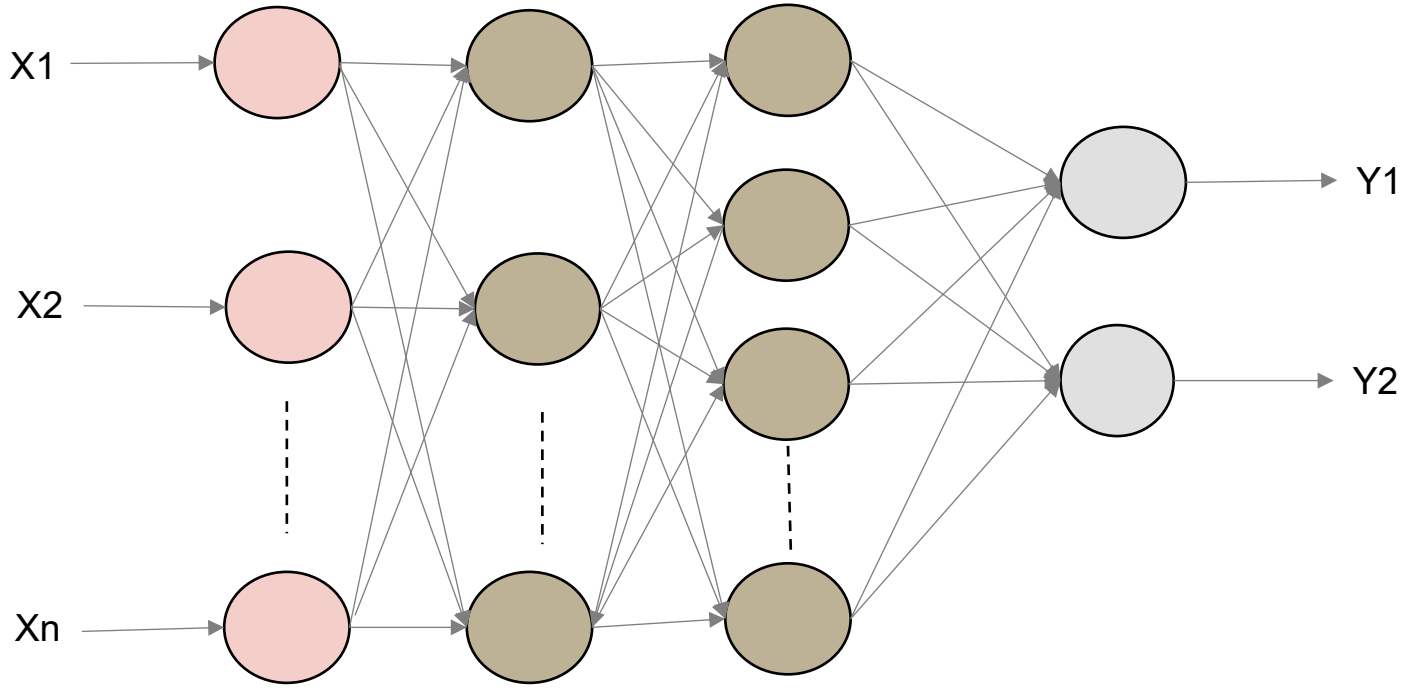
Apply

Think, plan
& reason

Problem

Solution

Multiple layer ANN Yi=f(Xi)



Inputs | Input Layer | Hidden Layers | Output Layer | Output/s

Example: Artificial Neural Network

Acquire → Learning

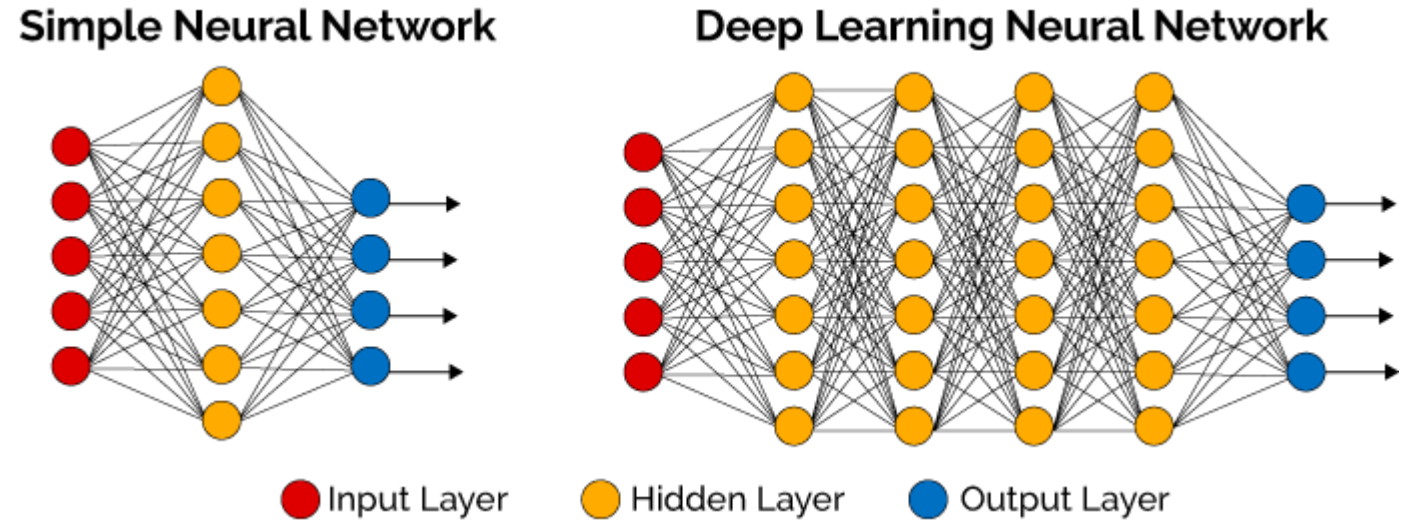
Memory
Adaptability

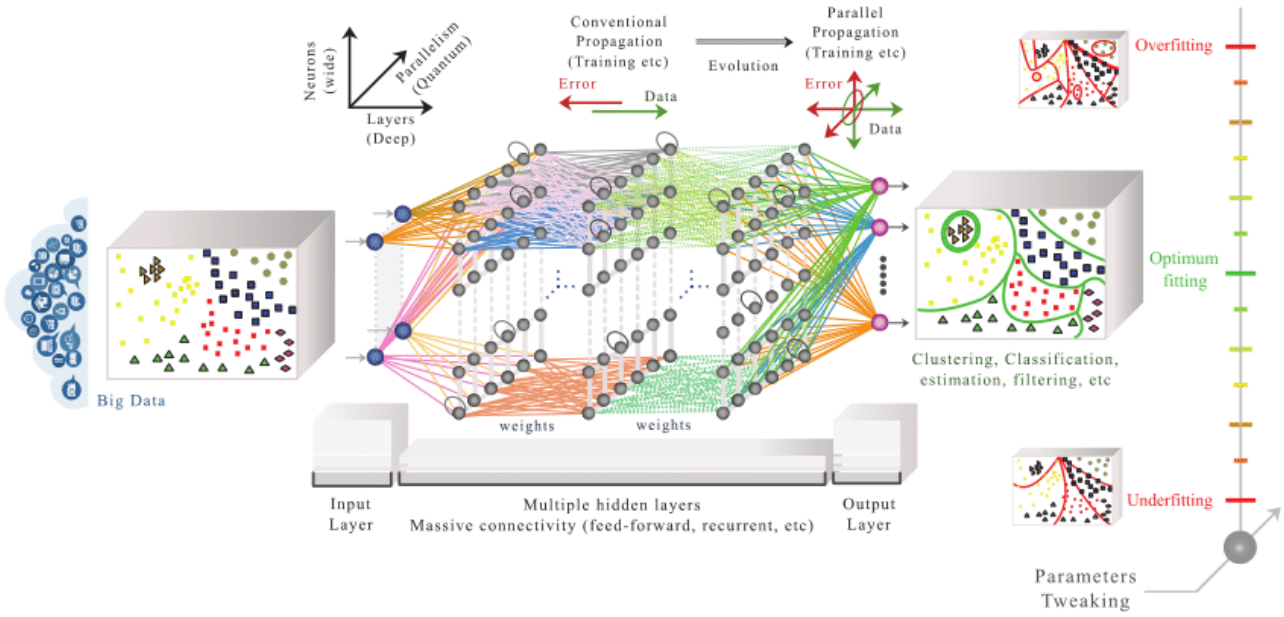
Tacit | Explicit | Implicit

Apply

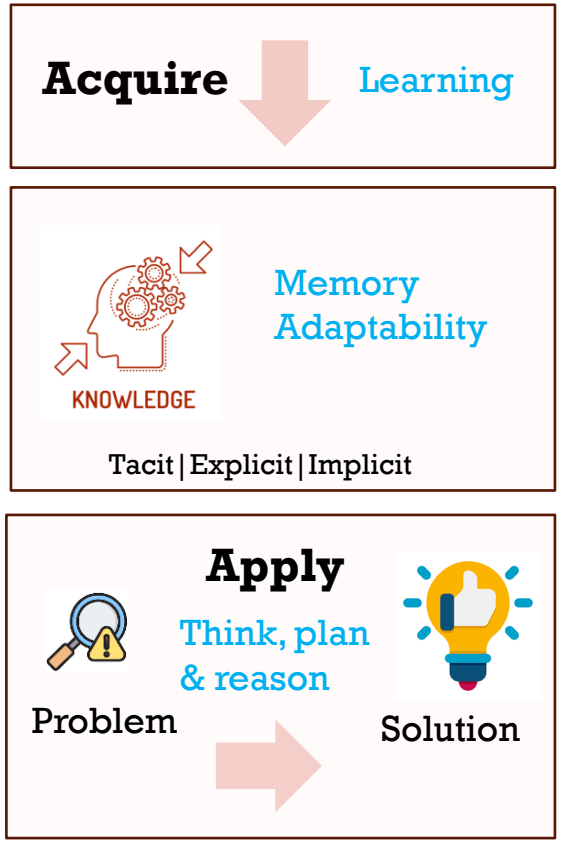
Think, plan & reason

Problem → Solution





Example: Neural networks



- Problem definition
- Problem classification
  - Prediction, Pattern Classification, Clustering/Categorization, Optimization, Control, Associative Memory etc.
- Selection of proper software (**off-the-shelf/Libraries-Open Source/programmable: e.g. matlab**)
- Data Collection
- Data preprocessing and presentation. Dividing data in training set, validation and test set
- Selection of neural network architecture
- Setting up parameters for network architecture
  - No Of Layers, number of neurons in hidden layer, learning algorithms, transfer function
- Setting training parameters and performance measures
  - No of iterations, exit/acceptance criteria, evaluation of test sets etc.
- Training the network up to expected performance by changing various parameters. This is **trial-and-error.**
- Testing the network
- Storing the network (neural network project) for live applications

Example: Neural networks: Training

Acquire → Learning

Memory
Adaptability

Tacit | Explicit | Implicit

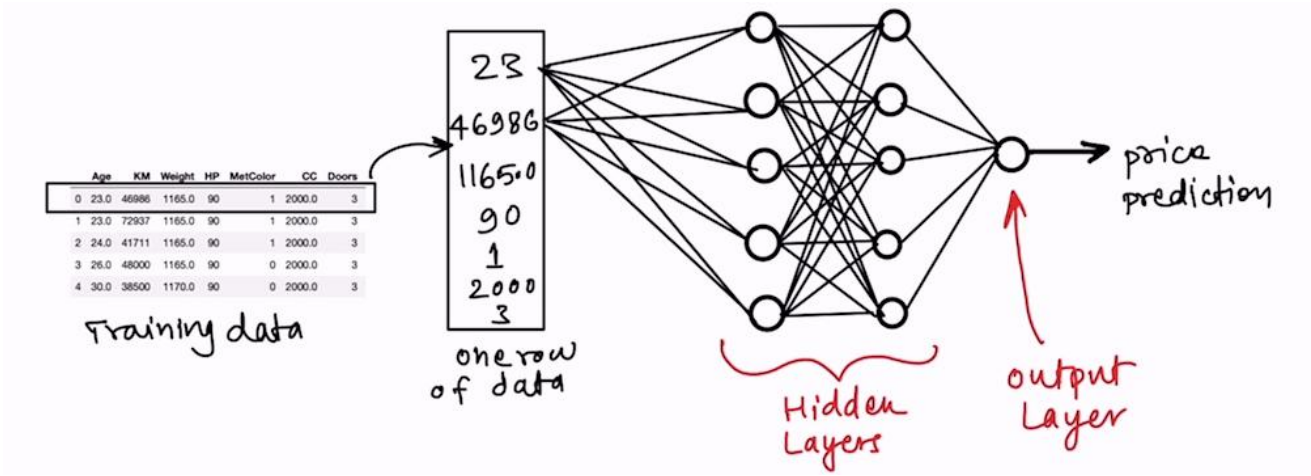
Apply

Think, plan & reason

Problem → Solution

- ✓ Data-set
- ✓ Architecture, algorithms, preprocessing etc.
- ✓ Topology
- ✓ Weights/Bias
- ✓ Activation functions

- ✓ Training (training data, test data, algorithms, batch size, epochs, learning rate, convergence, etc.)
- ✓ Trial n error
- ✓ Results
- ✓ Overfitting



Example: Neural networks: Training

Acquire → Learning

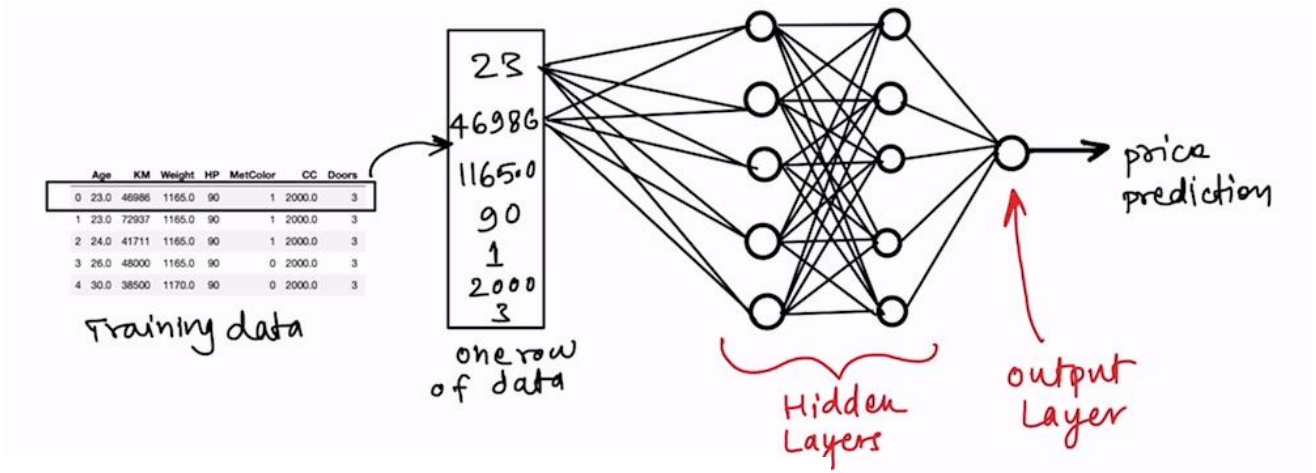
Memory
Adaptability

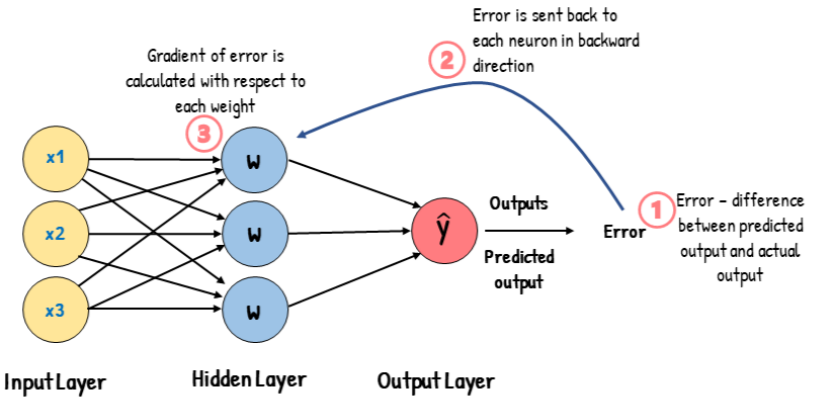
Tacit | Explicit | Implicit

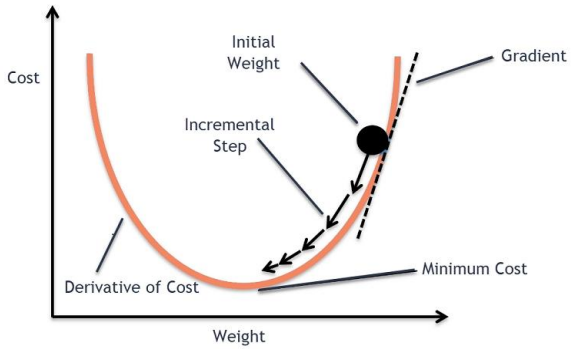
Apply

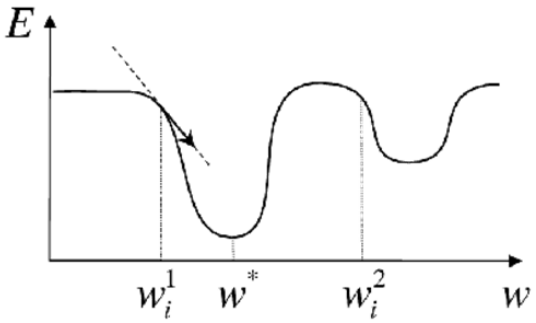
Think, plan & reason

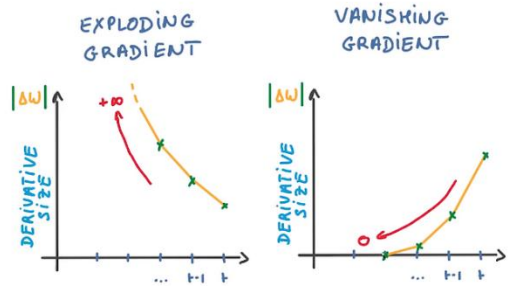
Problem → Solution











Example: Neural networks: Training

Acquire → Learning

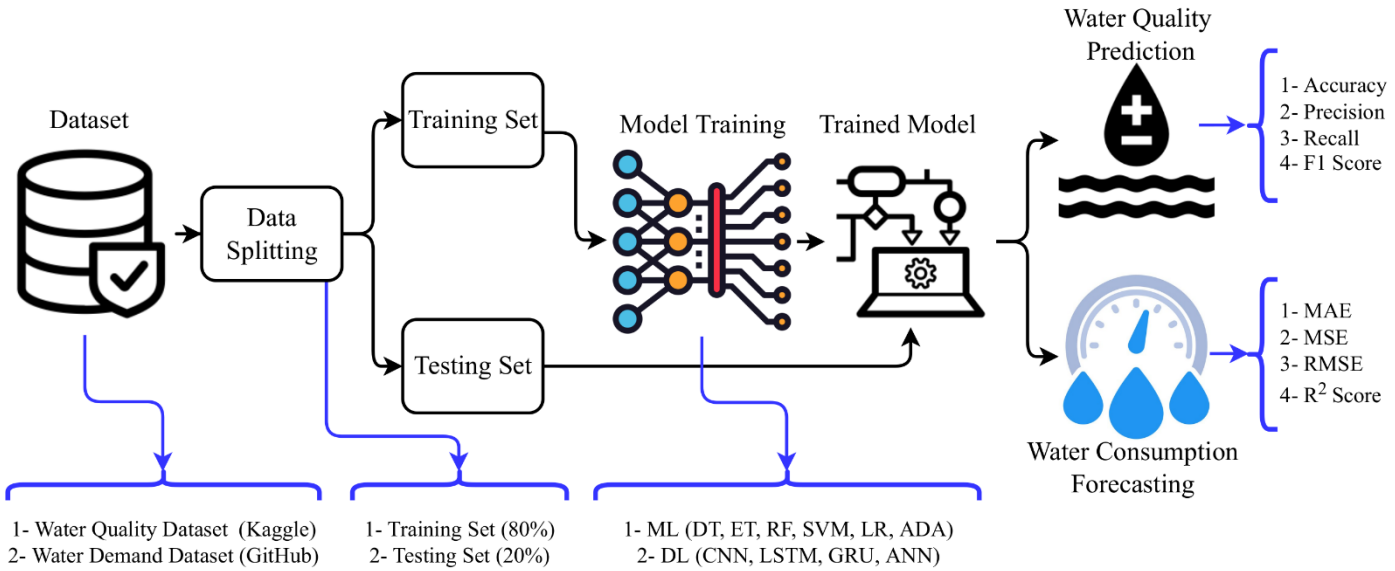
Memory
Adaptability

Tacit | Explicit | Implicit

Apply

Think, plan & reason

Problem → Solution



Example: An Artificial Neural Network Model for Water Quality and Water Consumption Prediction

Example: Neural networks: Training: Example

Acquire → Learning

Memory
Adaptability

Tacit | Explicit | Implicit

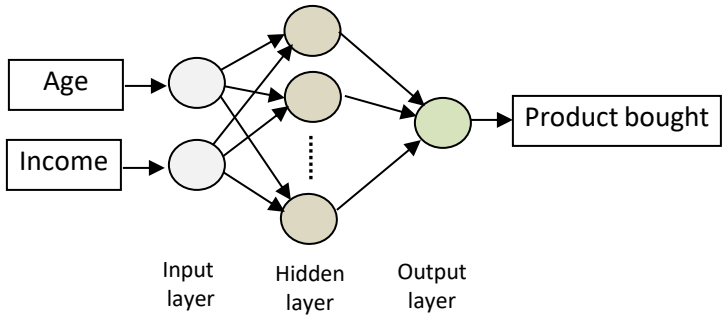
Apply

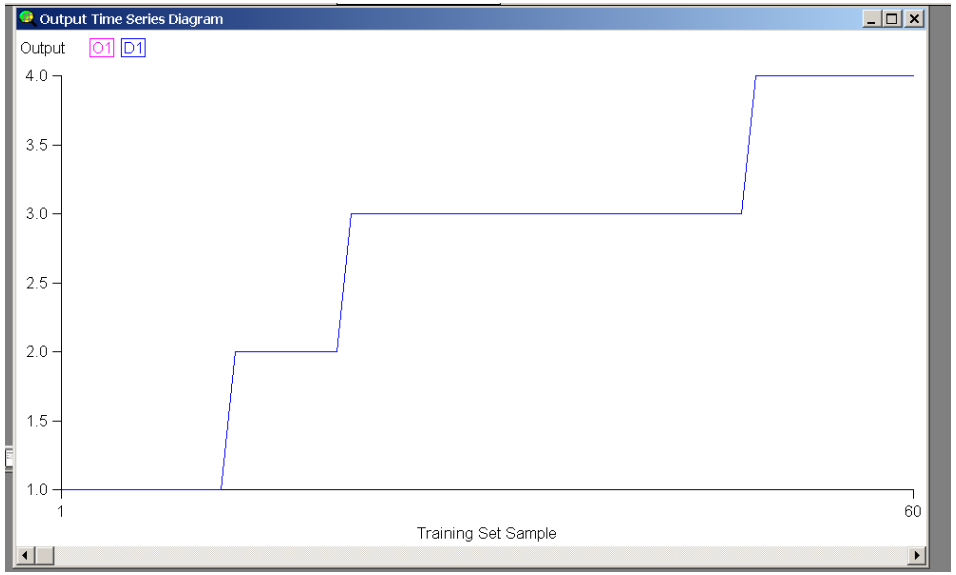
Think, plan & reason

Problem → Solution

Classification (sample)

Age	Income	Income_Group	Product
28	20000	3	4
45	23000	3	3
19	10000	1	2
24	14000	1	2
40	20000	3	3
39	23000	3	3
21	10000	1	2
28	25000	4	4
27	21000	3	4
35	25000	4	3
36	15000	2	1
38	30000	5	3
40	25000	4	3
42	14000	1	1
50	30000	5	3
28	21000	3	4
29	20000	3	4
34	29000	4	3
31	30000	5	3
24	15000	2	2
23	22000	3	4
41	17000	2	1





Example: Neural networks: Training: Example

Acquire → Learning

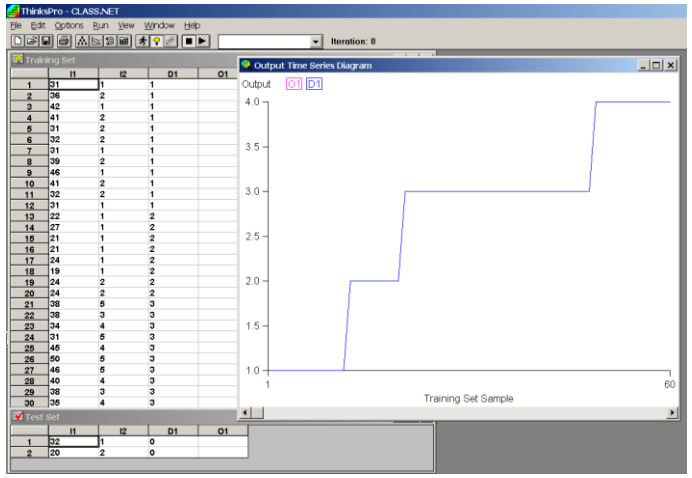
Memory
Adaptability

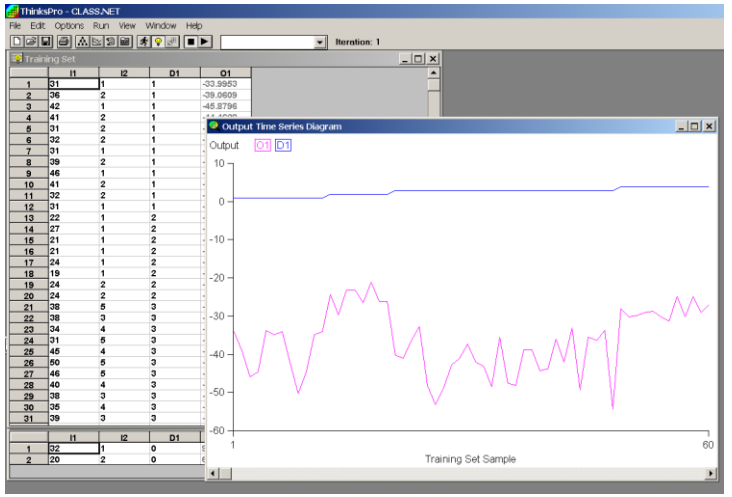
Tacit | Explicit | Implicit

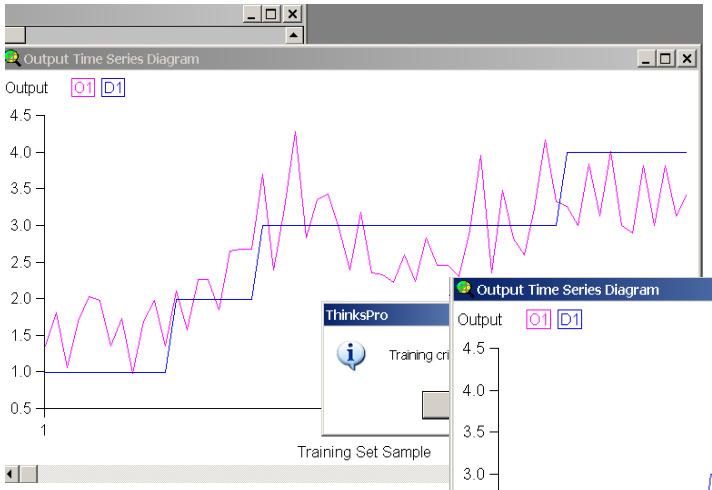
Apply

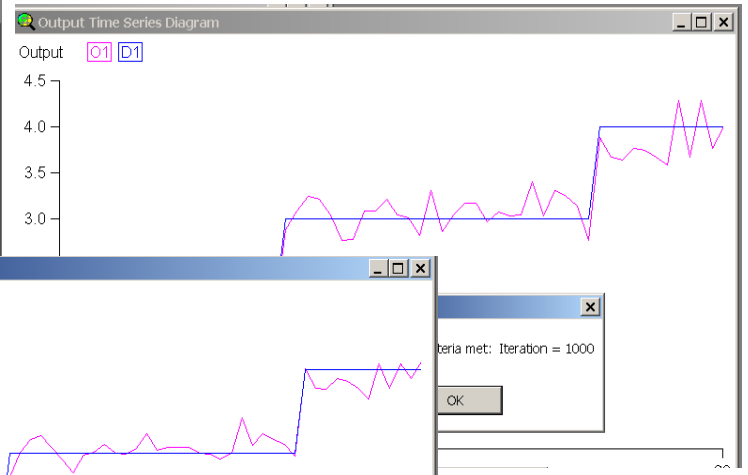
Think, plan & reason

Problem → Solution









Example: Neural networks: Sample problem types

Acquire → Learning

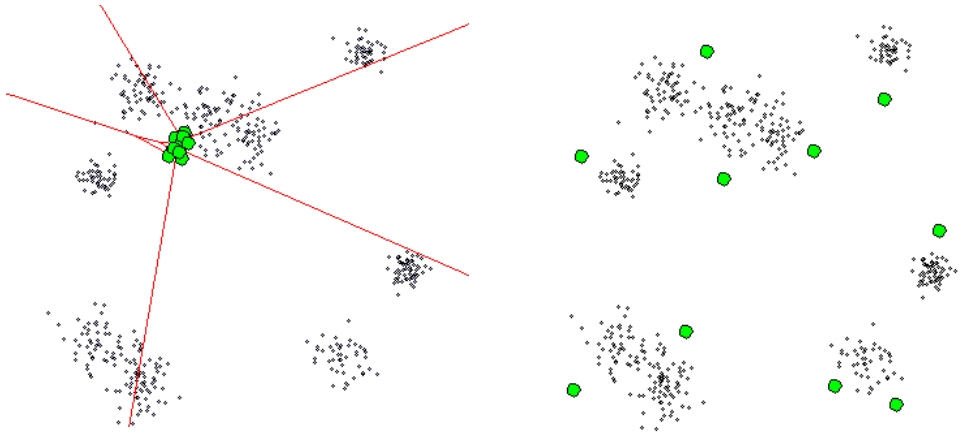
Memory
Adaptability

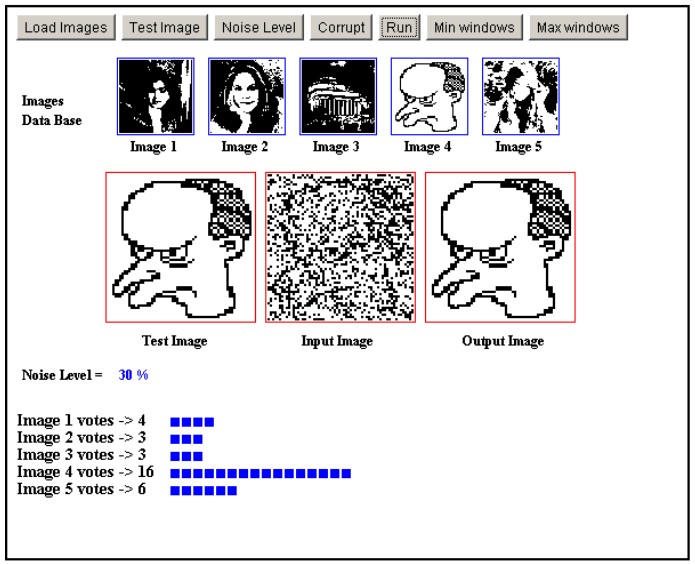
Tacit | Explicit | Implicit

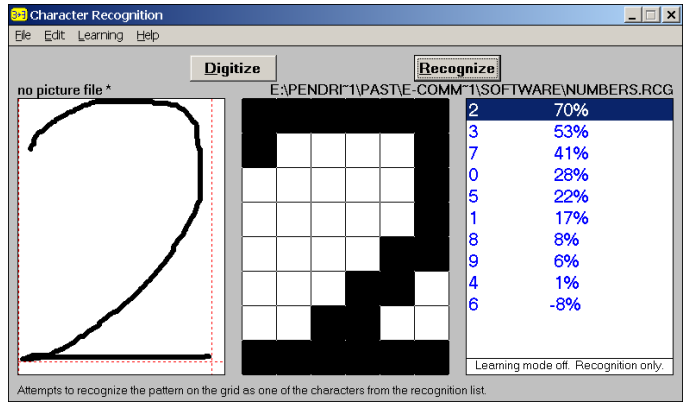
Apply

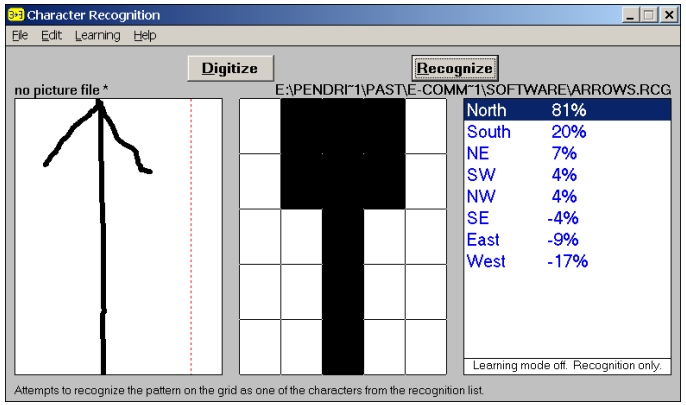
Think, plan & reason

Problem → Solution









Example: Neural networks: Training

Acquire → Learning

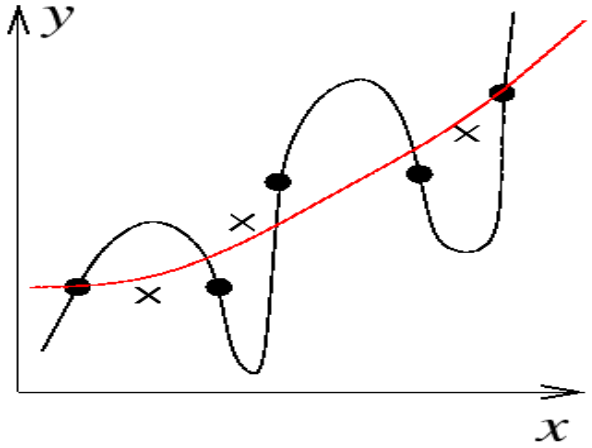
Memory
Adaptability

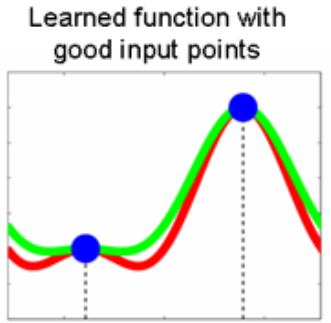
Tacit | Explicit | Implicit

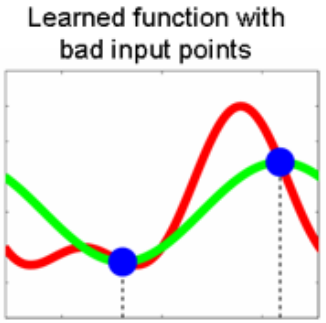
Apply

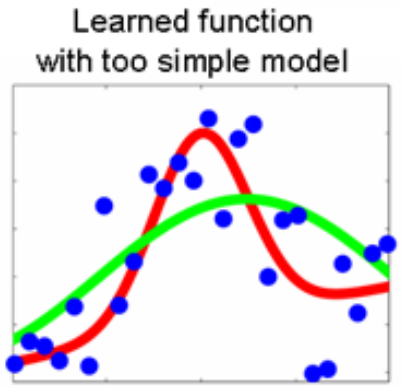
Think, plan & reason

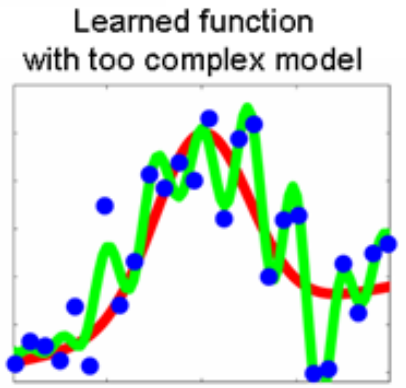
Problem → Solution

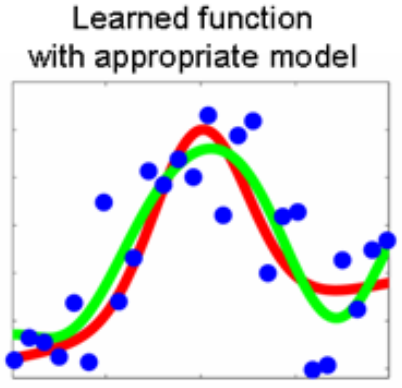




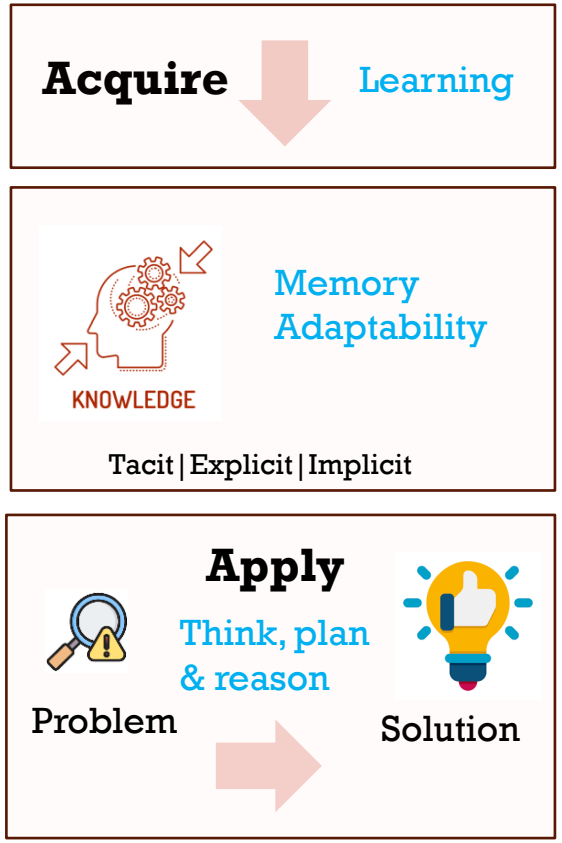








Example: Characteristics of ANNs



- Not programmed because they **learn** by example.
- Can **generalize** from their training data to other 'new' data. That is, the ability to interpolate from a previous learning experience. e.g. broomstick example.
- **Fault tolerant**. That is, they can produce a reasonably correct output from noisy and incomplete data, whereas conventional computers usually require correct data.
- **Fast** because many interconnected processing units work in parallel.
- On being **damaged**, they degrade gracefully unlike conventional hardware/software which simply fails to work.
- Relatively inexpensive to build, but **computationally intensive** to train.
- Particularly suited to problems whose solution is complex and difficult to specify, but which provides an abundance of data from which a response can be learnt.
- Can be trained to generate non-linear mappings i.e. work on real world problems! For example, predicting the weather forecast.
- **Adaptability** to new examples

Example: Types of ANNs

Acquire → Learning

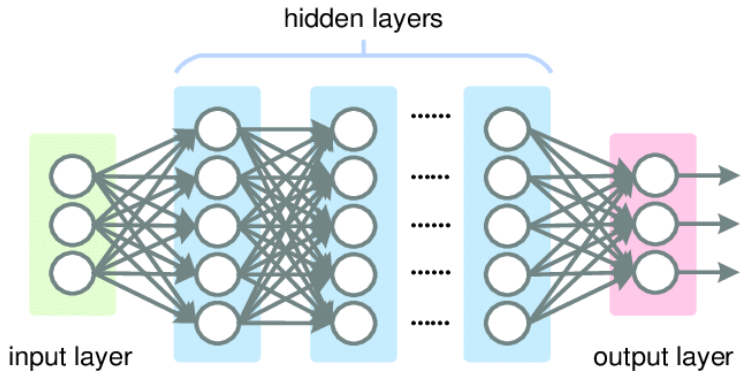
Memory
Adaptability

Tacit | Explicit | Implicit

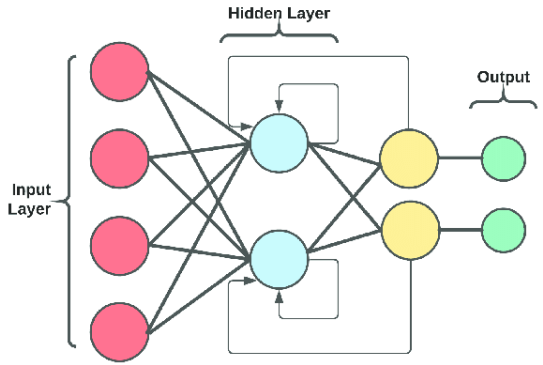
Apply

Think, plan & reason

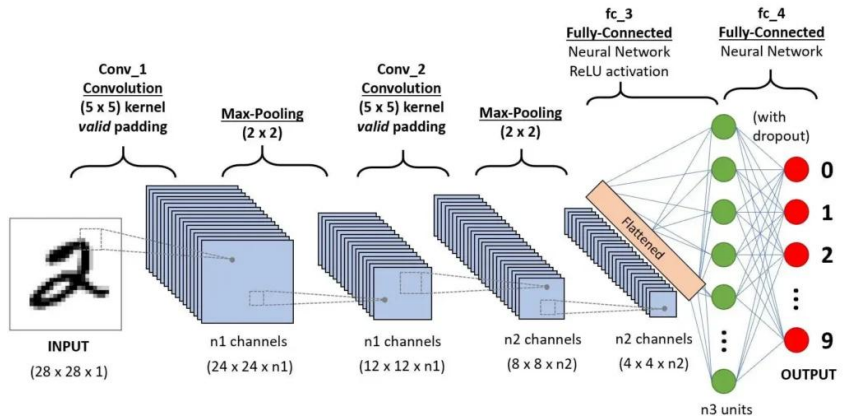
Problem → Solution



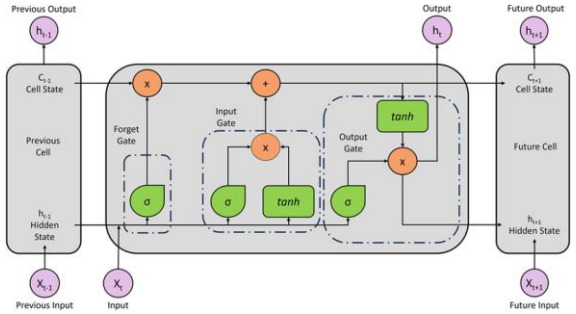
Fully feed-forward networks (FFN)



Recurrent neural network (RNN)



Convolutional Neural Networks (CNNs)



Long Short Term Memory Network (LSTM)

Example: Types of ANNs

Acquire → Learning

Memory
Adaptability

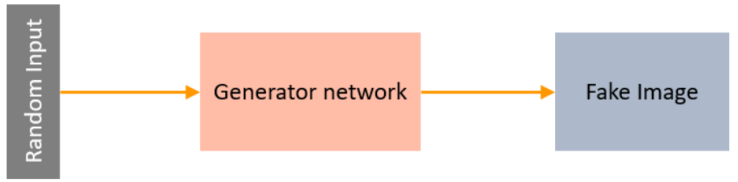
Tacit | Explicit | Implicit

Apply

Think, plan & reason

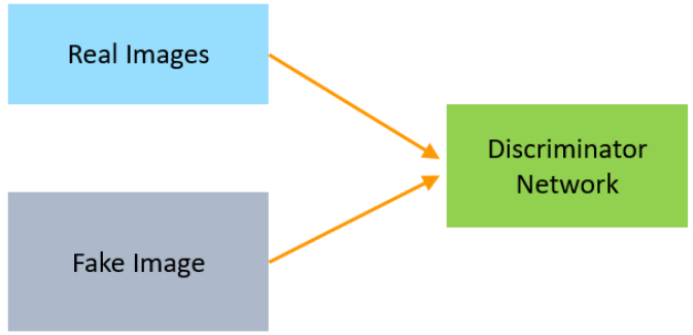
Problem → Solution

Generative Adversarial Networks



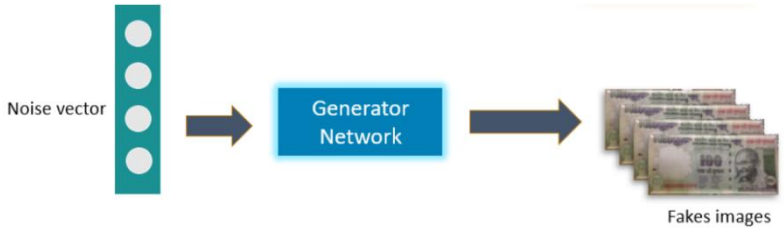
Generator

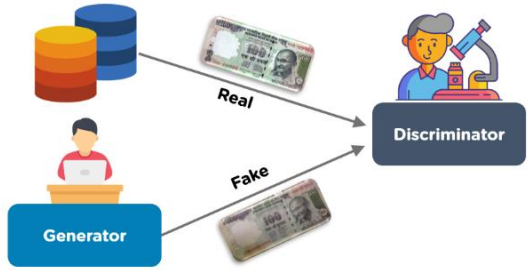
The main aim of the Generator is to make the discriminator classify its output as real.



Discriminator

The discriminator classifies both real data and fake data from the generator.





Example: Types of ANNs

Acquire → Learning

Memory
Adaptability

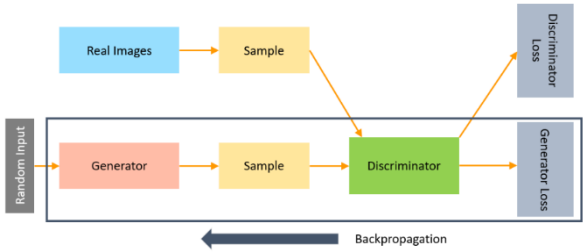
Tacit | Explicit | Implicit

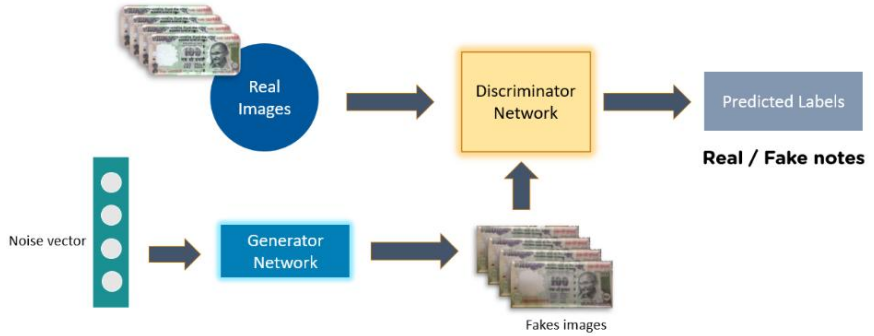
Apply

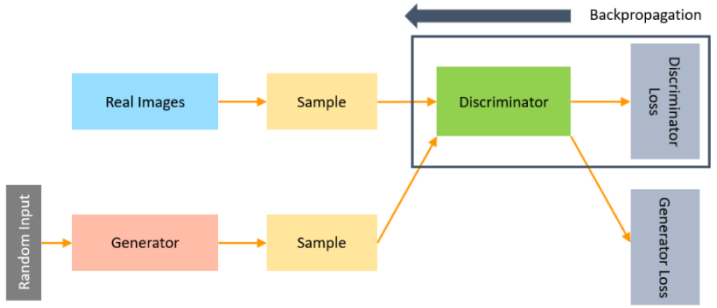
Think, plan & reason

Problem → Solution

Generative Adversarial Networks







1. Define the problem
2. Choose the architecture of GAN
3. Train discriminator on real data
4. Generate fake inputs for the generator
5. Train discriminator on fake data
6. Train generator with the output of the discriminator

Reinforcement Learning

Acquire → Learning

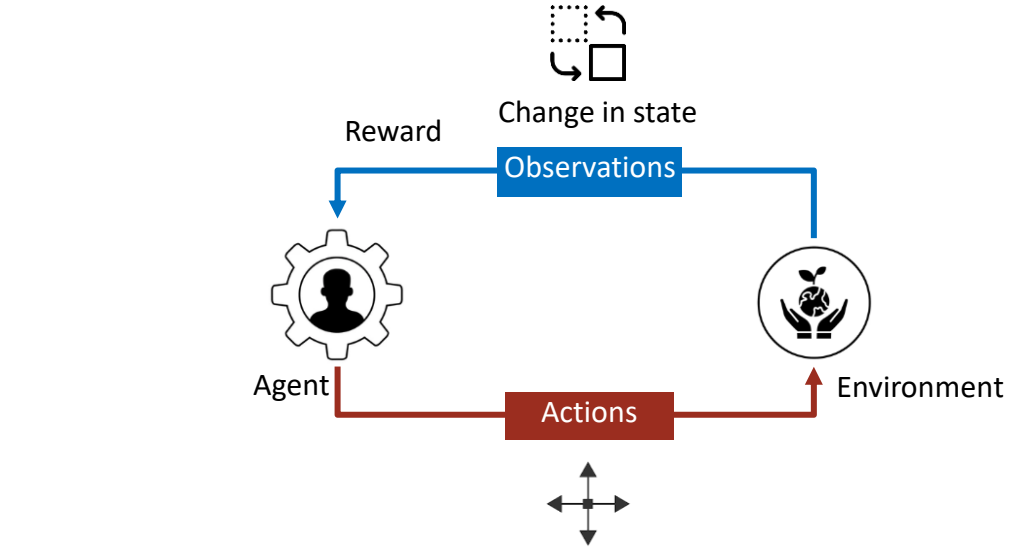
Memory
Adaptability

Tacit | Explicit | Implicit

Apply

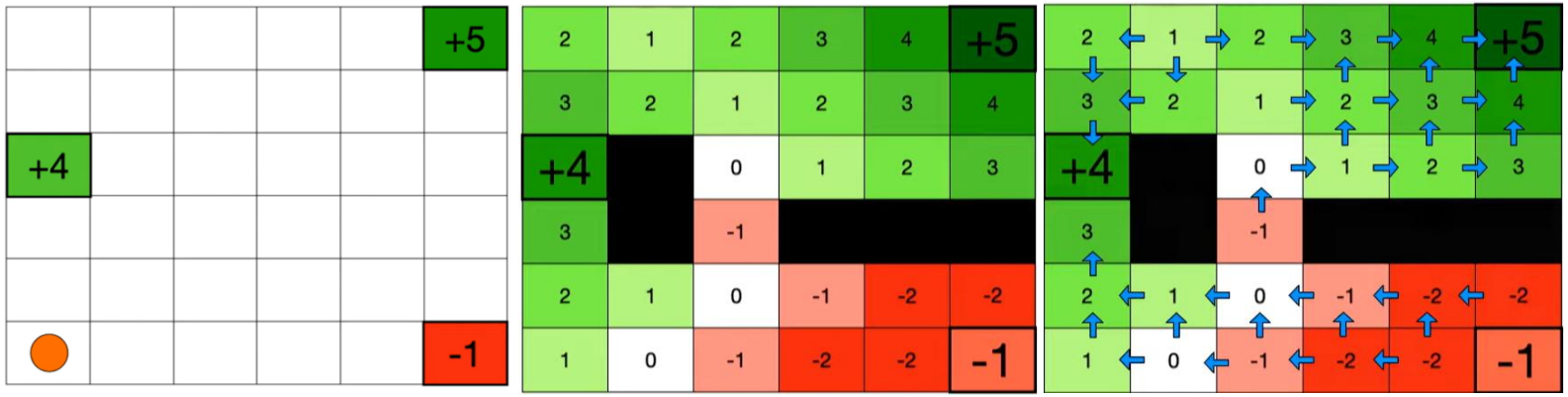
Think, plan & reason

Problem → Solution



Q function: Used to evaluate the expected rewards of taking a particular action in a given state. Captures expected total future reward an agent in a state can receive in future by executing certain action.

Agent takes best action which returns highest reward



[A friendly introduction to deep reinforcement learning, Q-networks and policy gradients]

Reinforcement Learning

Acquire → Learning

Memory
Adaptability

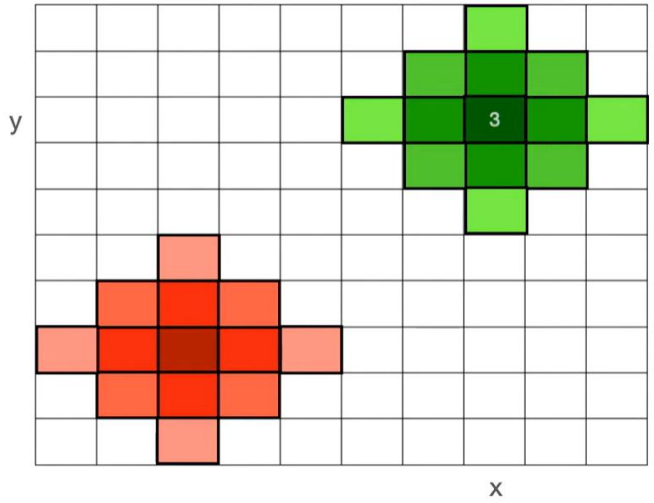
Tacit | Explicit | Implicit

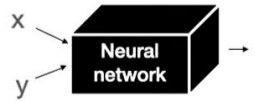
Apply

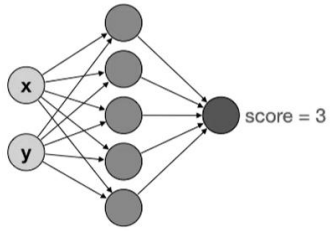
Think, plan & reason

Problem → Solution

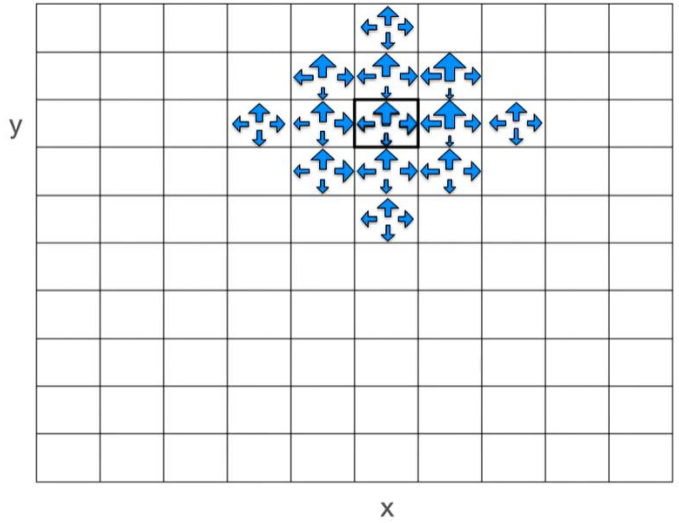
Value network

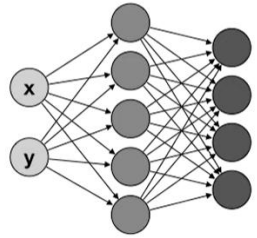






Policy network





[A friendly introduction to deep reinforcement learning, Q-networks and policy gradients]

Reinforcement Learning: Example

Acquire Learning

Memory
Adaptability

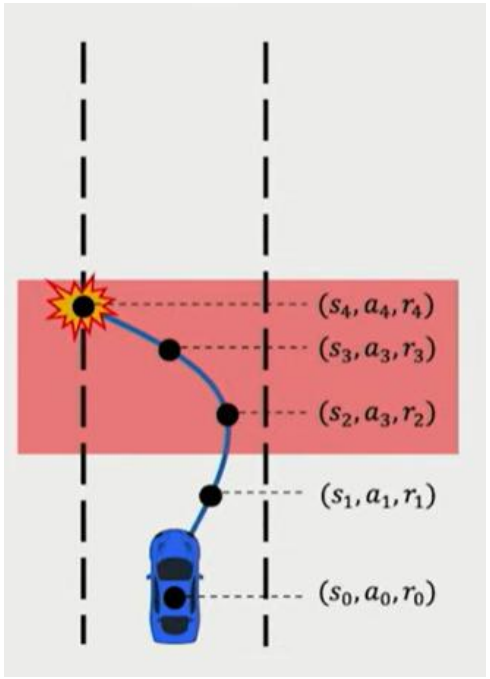
Tacit | Explicit | Implicit

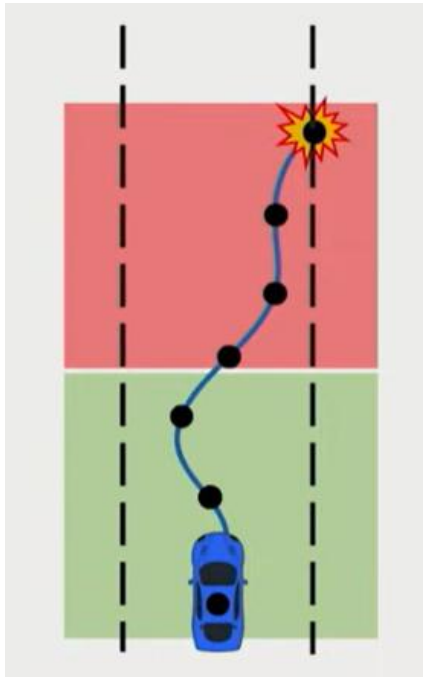
Apply

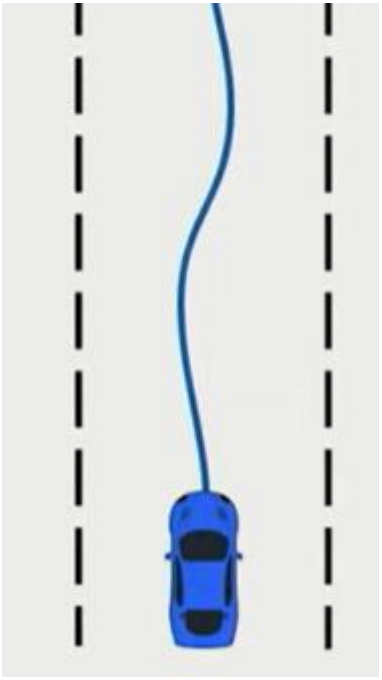
Think, plan & reason

Problem

Solution







Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

Learning Autonomous Driving in Simulation

Reinforcement Learning: Example

Acquire → Learning

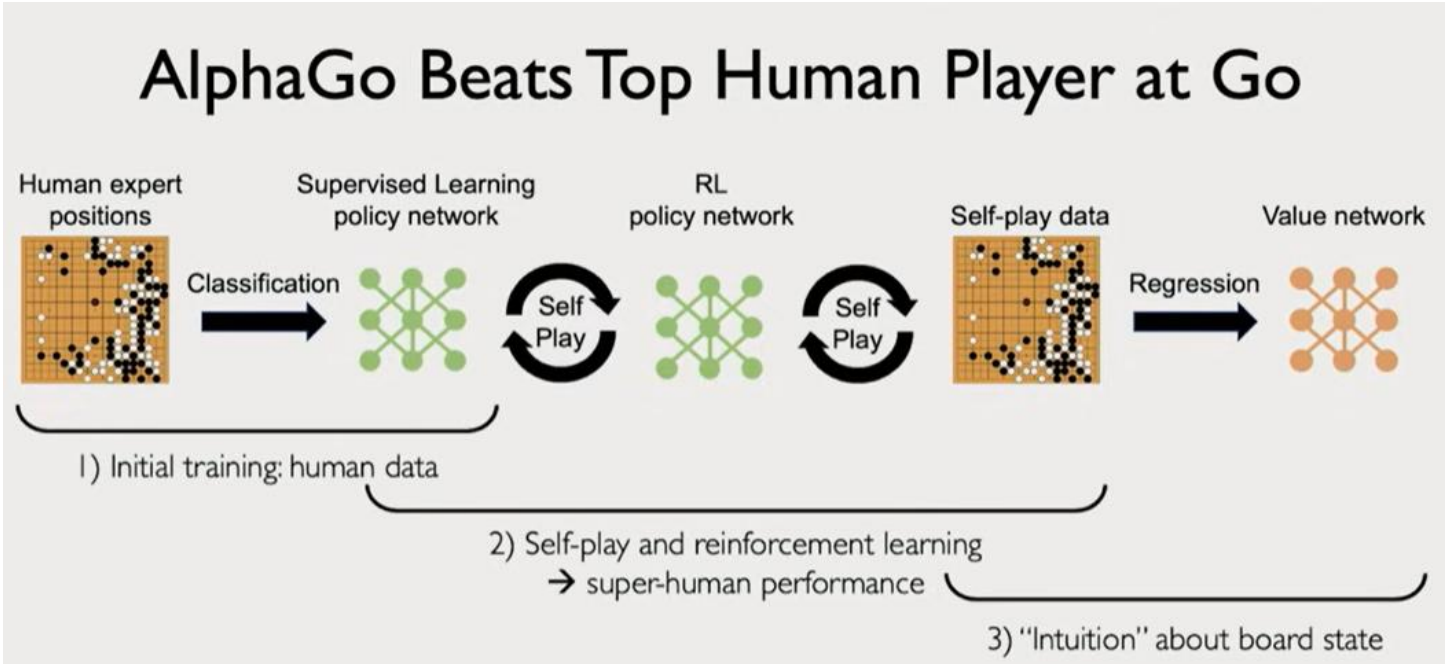
Memory
Adaptability

Tacit | Explicit | Implicit

Apply

Think, plan & reason

Problem → Solution



Example: Types of ANNs

Acquire → Learning

Memory
Adaptability

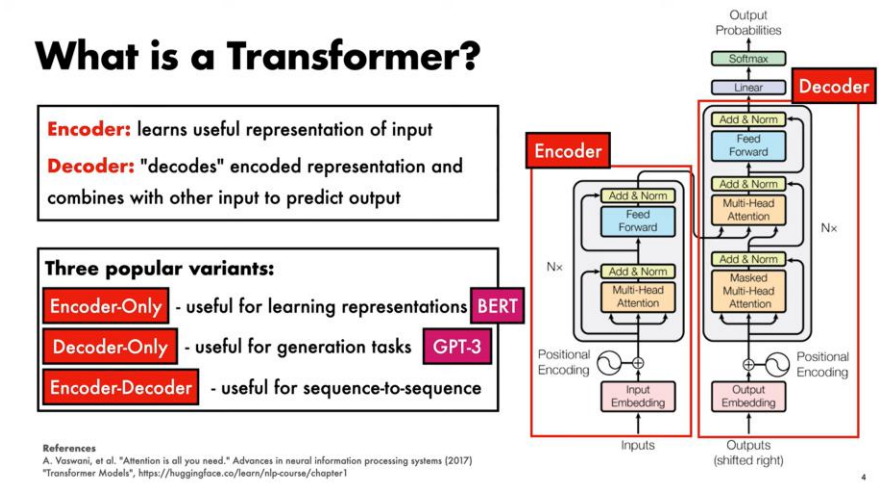
Tacit | Explicit | Implicit

Apply

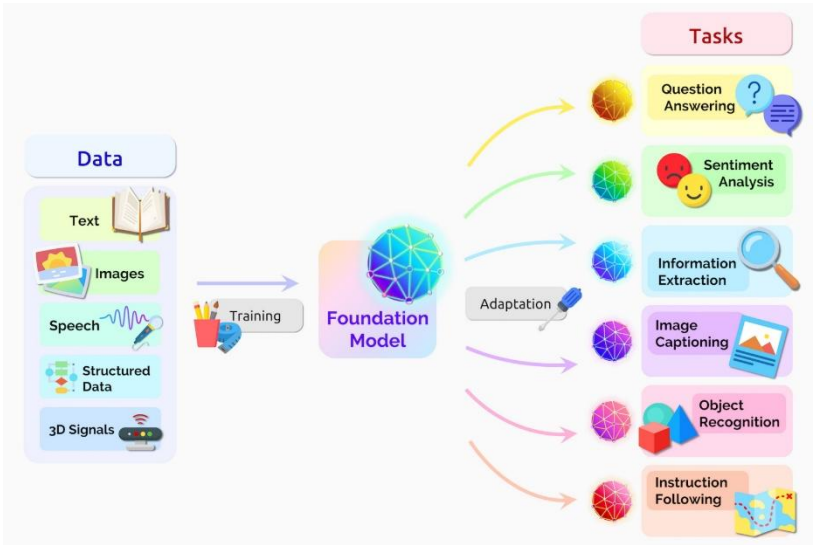
Think, plan & reason

Problem → Solution

Transformers



https://www.youtube.com/watch?v=vsqKGZT8Qn8
https://www.youtube.com/watch?v=ZhAz268Hdpw



https://blogs.nvidia.com/blog/what-is-a-transformer-model/

Example: Types of ANNs

Acquire → Learning

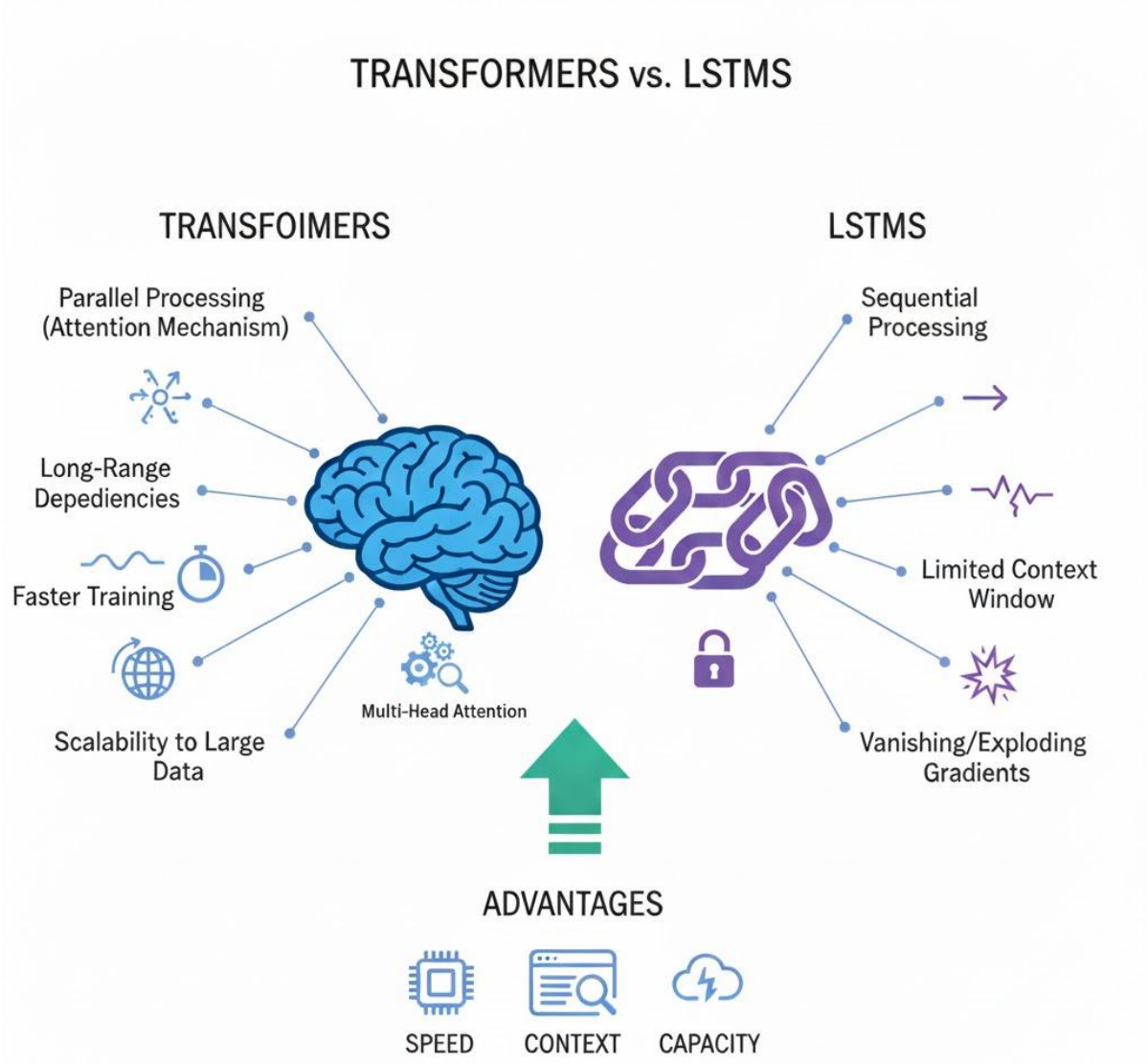
Memory
Adaptability

Tacit | Explicit | Implicit

Apply

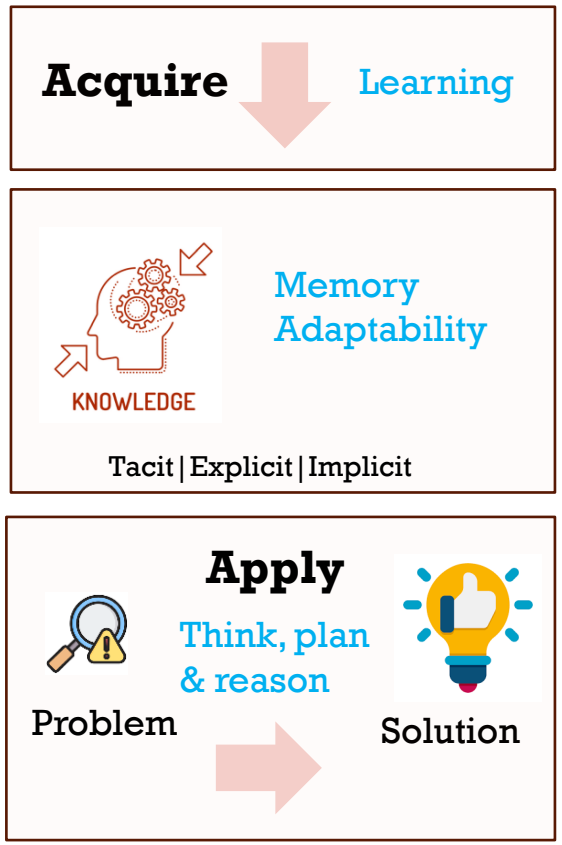
Think, plan & reason

Problem → Solution



https://blogs.nvidia.com/blog/what-is-a-transformer-model/

Example: ANN Issues and limitations



Issue Category	What is the Problem?	Why it is a Concern
Interpretability / Explainability	ANN decisions are opaque ("black box")	Difficult to justify decisions in regulated domains (banking, insurance, government)
Data Dependency	Requires large volumes of high-quality labeled data	Costly data collection; poor performance in low-data or niche domains
Overfitting	Model memorizes training data instead of learning patterns	Performs well in training but fails in real-world scenarios
Computational Cost	High training and inference resource requirements (GPU/TPU, power)	Expensive infrastructure; limits scalability and edge deployment
Logical Reasoning	Weak at explicit rules, constraints, and symbolic logic	Poor fit for policy enforcement, workflows, and compliance systems
Debugging Difficulty	No clear way to trace errors to specific causes	Root-cause analysis is hard; retraining replaces debugging
Hyperparameter Sensitivity	Performance depends on tuning learning rate, layers, activation functions	Trial-and-error development; requires deep expertise
Incremental Learning	New training overwrites old knowledge (catastrophic forgetting)	Difficult to add new rules or exceptions safely
Bias & Fairness	Learns and amplifies bias present in training data	Ethical, legal, and reputational risks
Auditability & Traceability	No explicit decision trail or rule history	Weak governance, compliance, and version comparison

https://blogs.nvidia.com/blog/what-is-a-transformer-model/

Input types

Acquire → Learning

Memory
Adaptability

Tacit | Explicit | Implicit

Apply

Problem

Think, plan & reason

Solution

One Time | Conversational | Contextual | Augmented | Multiple | Orchestrated | Noisy | Uncertainty

Solution outcome

Acquire Learning

Memory
Adaptability

Tacit | Explicit | Implicit

Apply

Think, plan
& reason

Problem

Solution

- Numerical Results
- Results with explanation
- Actionable
- Recommendations
- Intermediate
- Real time | Deferred
- Integrated
- Personalized

Memory

Acquire → Learning

Memory Adaptability

Tacit | Explicit | Implicit

Apply

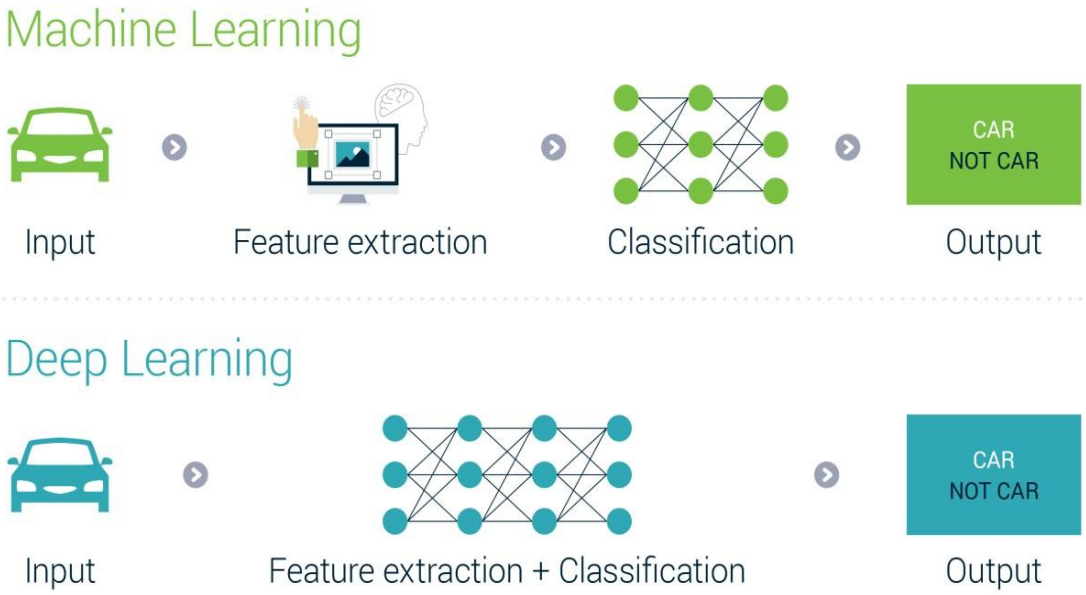
Problem

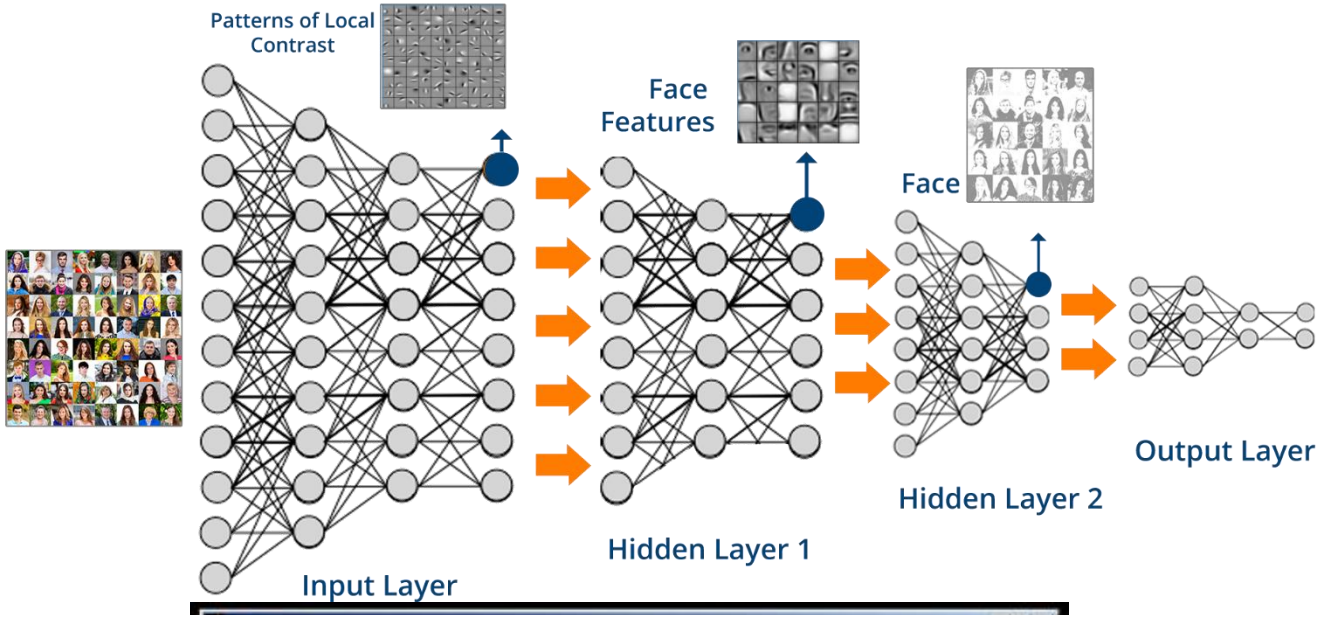
Think, plan & reason

Solution

- Models
- Session based
- Meta-data
- Context (facts)
- In-memory

ML and Deep ML

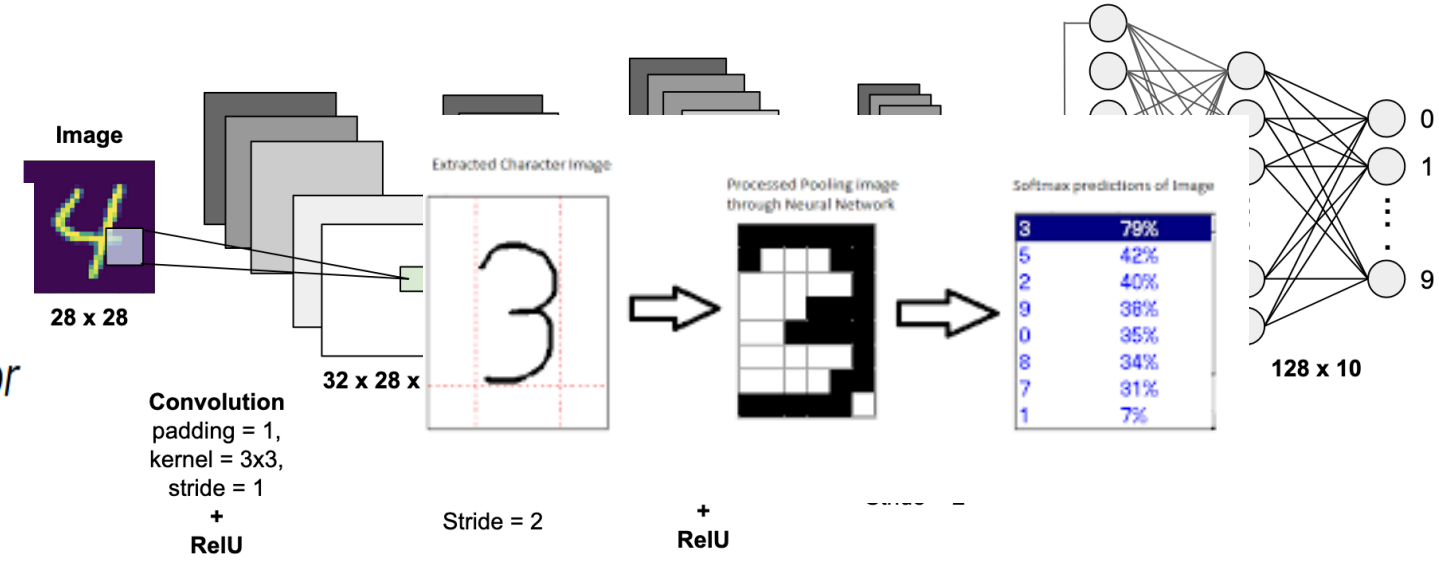




https://www.inteliment.com/blog/our-thinking/lets-understand-the-difference-between-machine-learning-vs-deep-learning/

CHARACTER RECOGNITION

Handwritten character Recognition: Training a Simple NN for classification using MATLAB



http://lab.fs.uni-lj.si/lasin/wp/IMIT_files/neural/doc/seminar5.pdf

Generative AI: Concepts and foundations: NLP

Tokenization (sentences/words)

```
from nltk.tokenize import sent_tokenize
text="Hello friends!. Good Morning! Today we will learn Natural Language Processing. It is very interesting"
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello friends!.', 'Good Morning!', 'Today we will learn Natural Language Processing.', 'It is very interesting']
```

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'friends', '!', '.', 'Good', 'Morning', '!', 'Today', 'we', 'will', 'learn', 'Natural', 'Language', 'Processing',
'.', 'It', 'is', 'very', 'interesting']
```

Stopwords Removal

```
text = "India is my country"
text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)
```

Part of speech tagging

```
from nltk import word_tokenize, pos_tag
s = "The name of our country is India"
print(pos_tag(word_tokenize(s)))
```

```
[('The', 'DT'), ('name', 'NN'), ('of', 'IN'), ('our', 'PRP$'), ('country', 'NN'), ('is', 'VBZ'), ('India', 'NNP')]
```

Stemming and Lemmatization

```
from nltk.stem import PorterStemmer
e_words= ["Played", "Playing", "Play"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

```
play
play
play
```

Synonyms and antonyms

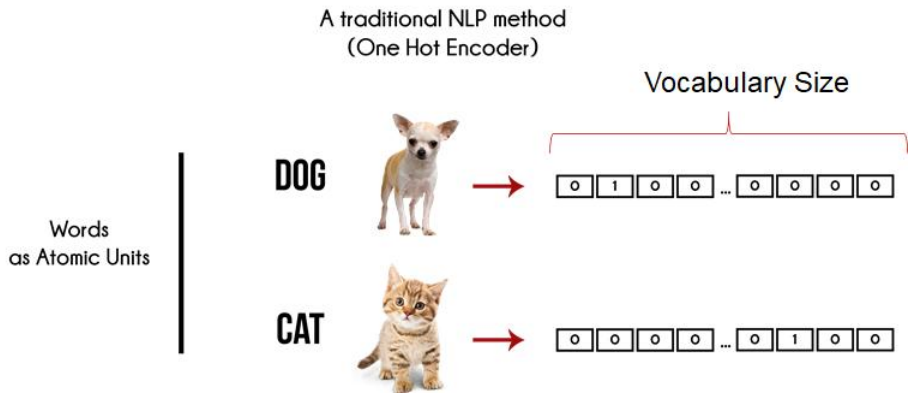
```
# look up top 6 words similar to 'polite'
w1 = ["polite"]
model.wv.most_similar (positive=w1,topn=6)
```

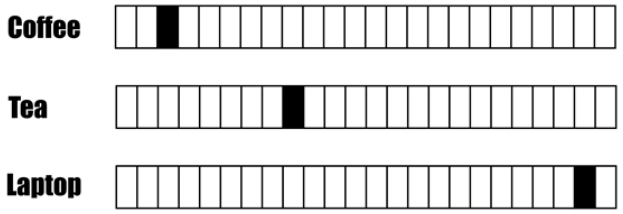
```
[('courteous', 0.9174547791481018),
 ('friendly', 0.8309274911880493),
 ('cordial', 0.7990915179252625),
 ('professional', 0.7945970892906189),
 ('attentive', 0.7732747197151184),
 ('gracious', 0.7469891309738159)]
```


Natural-language-processing-nlp-simplified-a-step-by-step-guide

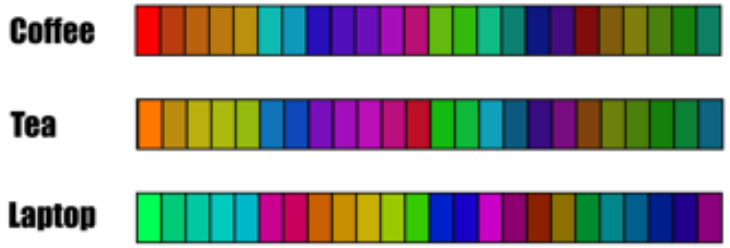
Generative AI: Concepts and foundations

Word Embeddings: numerical presentations of text





Connecting words: context (I painted it with **blue** colour, she painted it with **green** colour), how many times they are coming together (they **laughed** at **joke**, she **laughed** hearing that **joke**), helps in: reducing number of words (he is **great**, she is **good**), semantic relations (he likes **fruit**, she likes **apple**)



https://predictivehacks.com/a-high-level-introduction-to-word-embeddings/

Generative AI: Concepts and foundations

Issues with traditional NLP

Hi, I am Rajendra, my friends Akash and Swati visited my home yesterday after long. He did excellent in maths, now a days busy in his job of CTO, loves apple phones while continuing his hobby of reading fiction books. She still loves cooking and making fruit cakes, especially, made with apple.

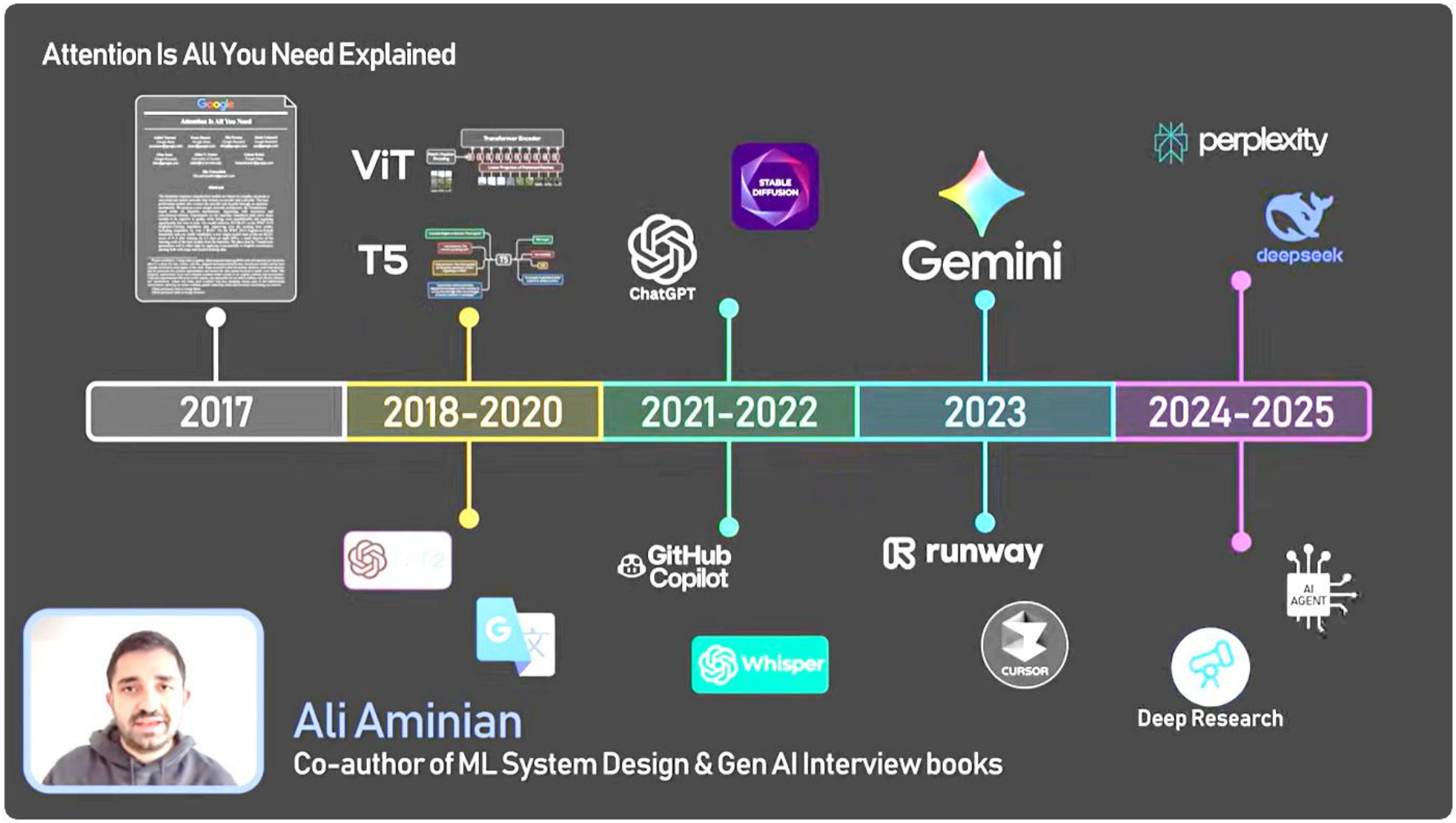
1. Limited sentences, missing continuity
2. Context, meanings and grammar
3. Sequential processing
4. …

Who are the friends of Rajendra?
Who is he about whom Rajendra talking about?
Who likes cooking? and what kind?
What does Akash do?
What does Swati like to cook?

…

Hi, I am Rajendra, my friends Akash and Swati visited my home yesterday after long. He did excellent in maths, now a days busy in his job of CTO, loves **apple** phones while continuing his hobby of reading fiction books. She still loves cooking and making fruit cakes, especially, made with **apple**.

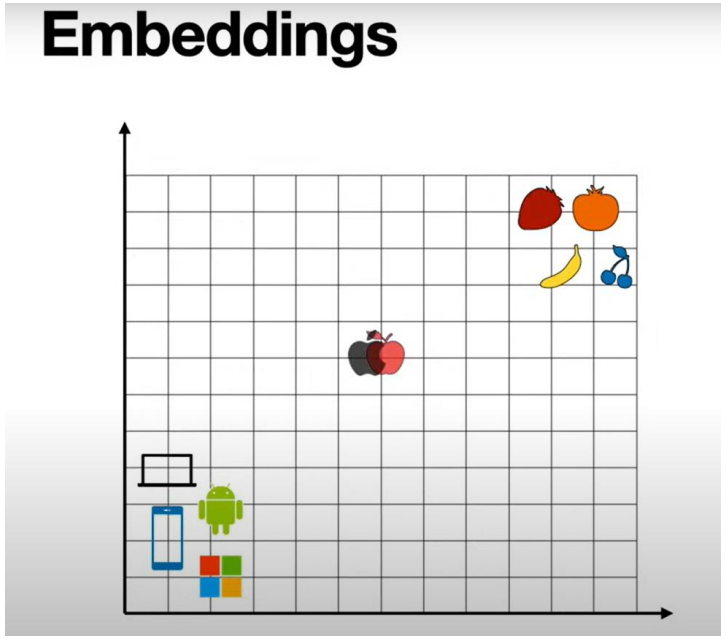
Generative AI: Concepts and foundations

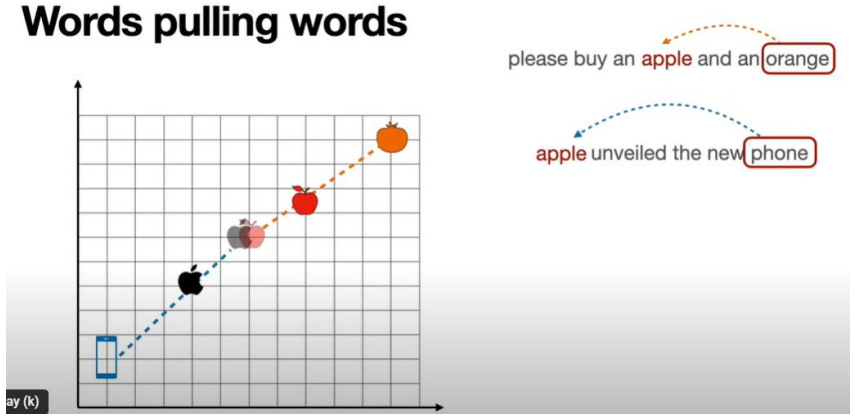


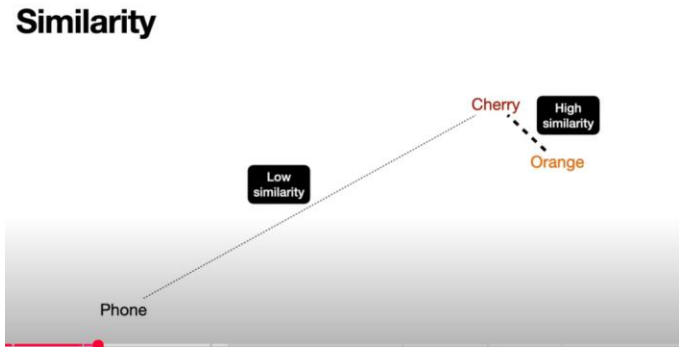
[Transformers Step-by-Step Explained (Attention Is All You Need)]

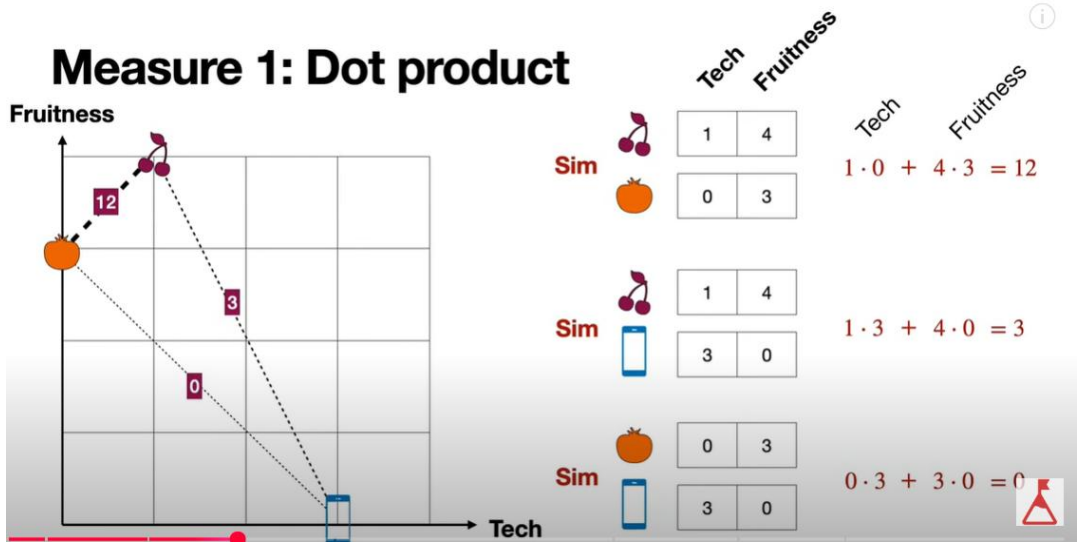
[How Attention Mechanism Works in Transformer Architecture]

Generative AI: Concepts and foundations: NLP



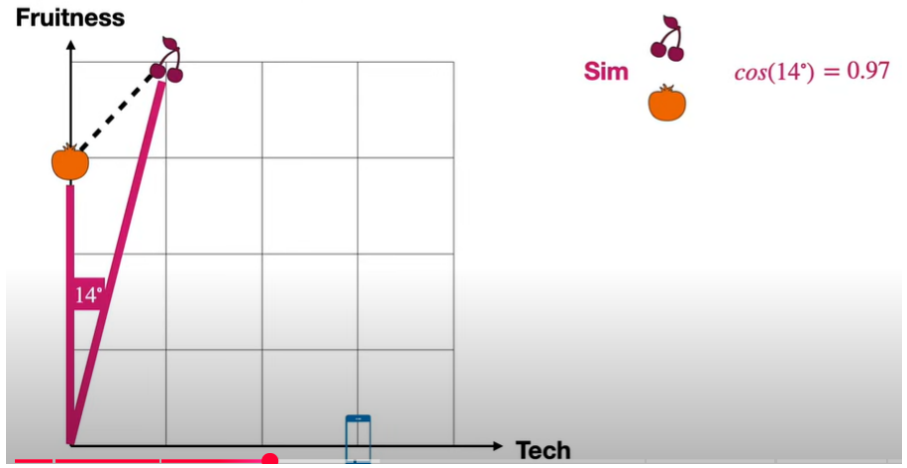




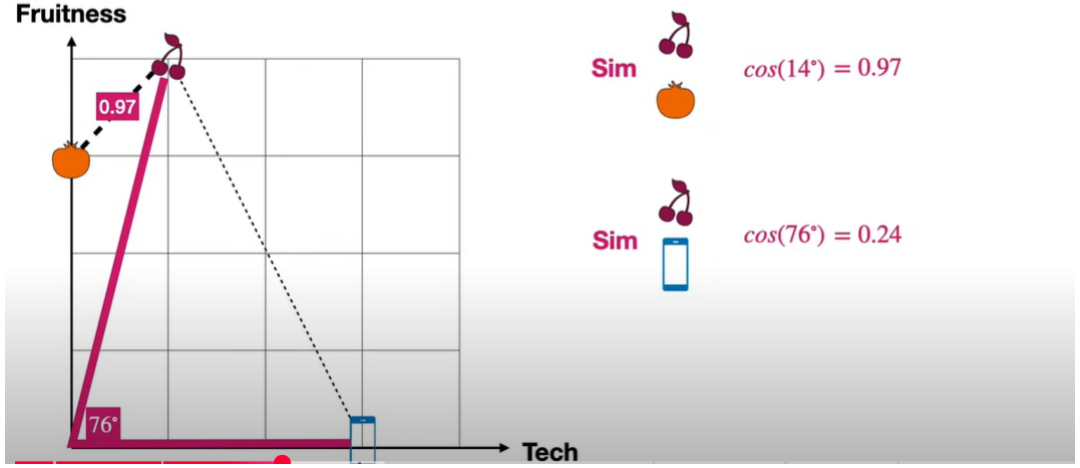


https://www.youtube.com/watch?v=UPtG_38Oq8o

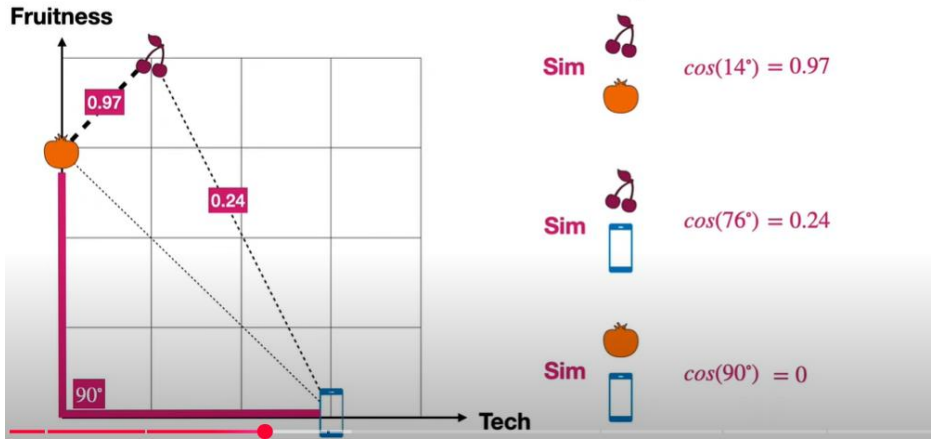
Measure 2: Cosine similarity



Measure 2: Cosine similarity

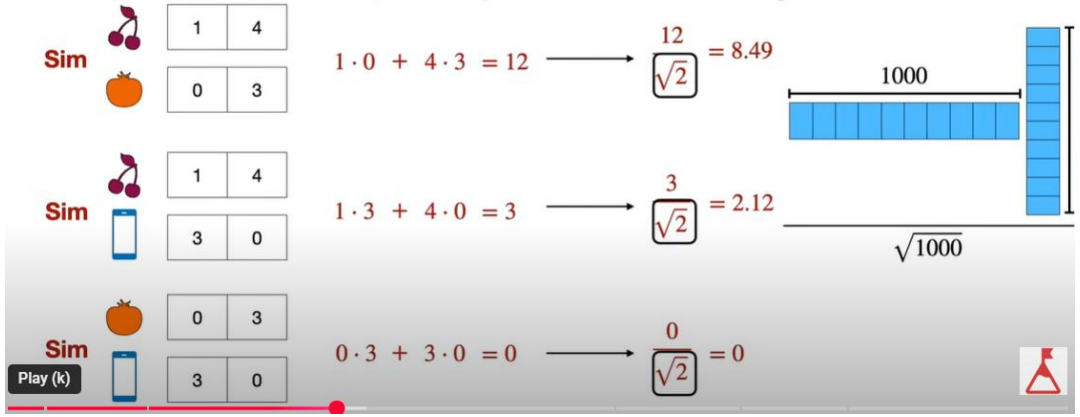


Measure 2: Cosine similarity

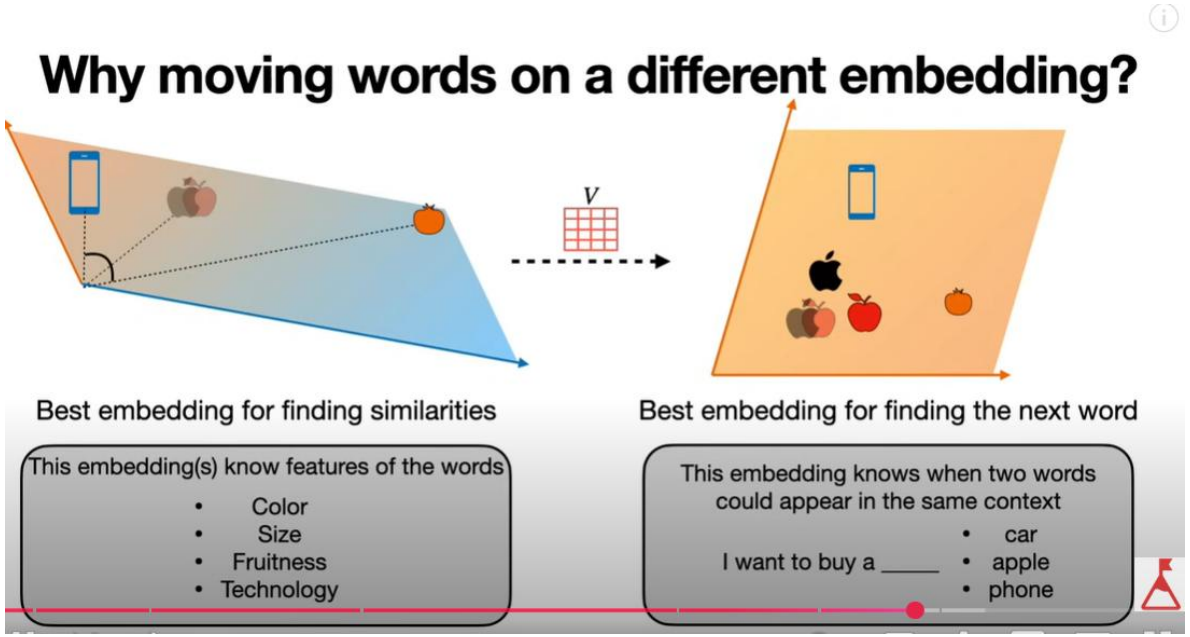


Measure 3: Scaled dot product

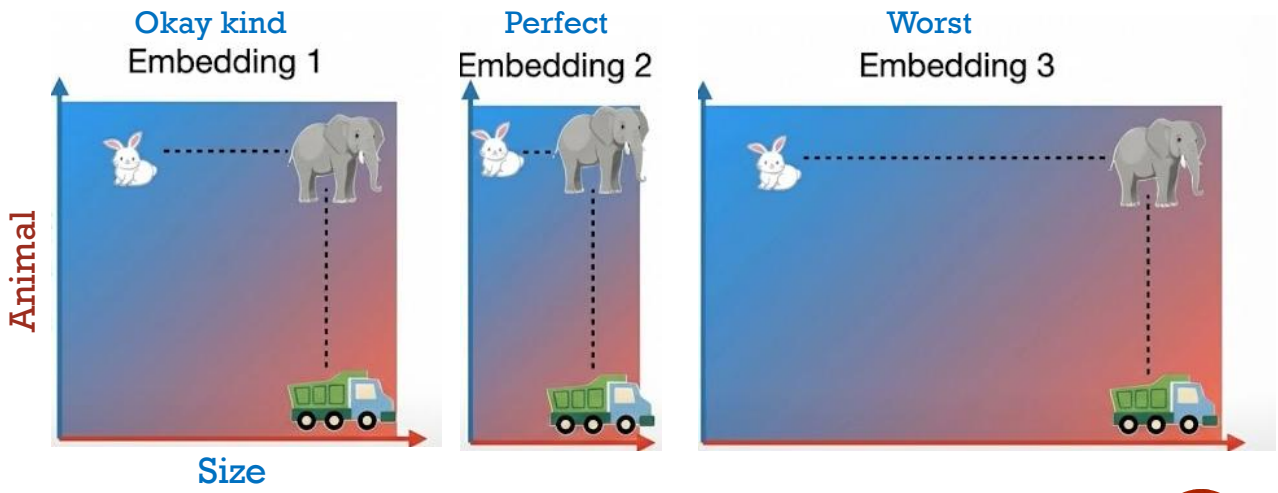
Dot product divided by the square root of the length of the vector



Generative AI: Concepts and foundations: NLP



Modelling semantics: Difference between rabbit and elephant is less than difference between truck and elephant when dimension size is taken into consideration. Means dimension animal should be considered at higher level than size.



https://www.youtube.com/watch?v=UPtG_38Oq8o

https://www.youtube.com/watch?v=RFdb2rKAqFw

Generative AI: Concepts and foundations: NLP

https://www.youtube.com/watch?v=bCz4OMemCcA

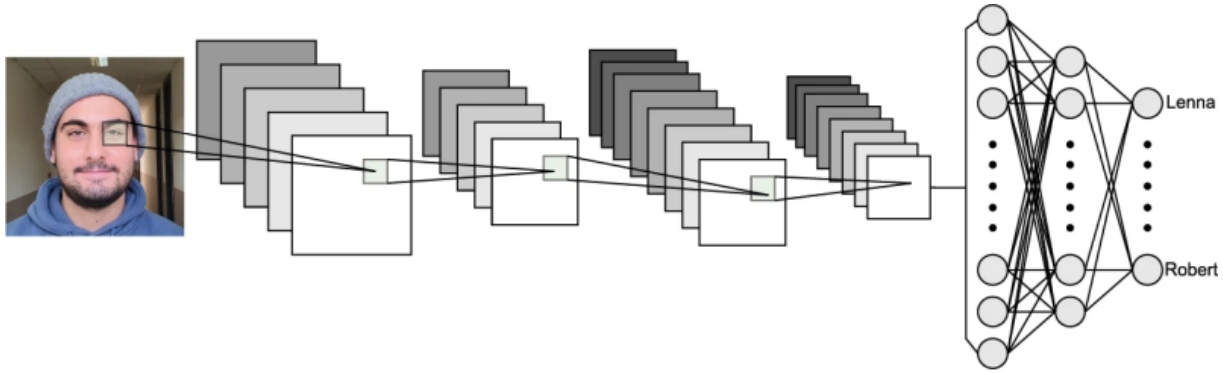
Generative AI: Concepts and foundations: NLP

Self-attention: allows models to relate words each other

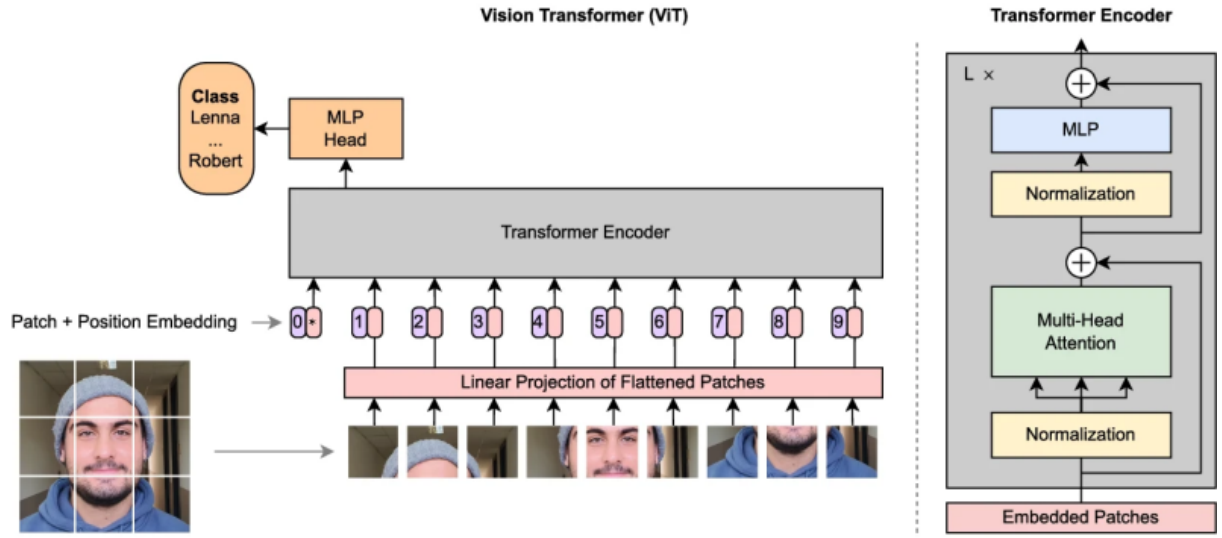
	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

https://www.youtube.com/watch?v=bCz4OMemCcA

Generative AI: Concepts and foundations: Vision Transformer



(a) Common CNN architecture



(b) Vision Transformer architecture

CNNs

PROS
- Efficient for spatial data
- Less data needed to train
- Just works well
- Local feature extraction

CONS
- Limited receptive field
- Less flexible
  - Different applications require different CNN architectures

ViTs

PROS
- Global context
- Scales incredibly well with
  - More data, more compute
- Flexible – one architecture

CONS
- Expensive to train
- Data hungry
- More complex

https://www.nature.com/articles/s41598-024-72254-w/figures/1

Generative AI: Concepts and foundations: NLP

Typical token embedding sizes by model scale

LLM Family	Model	Hidden Dimension (d_model)
OpenAI GPT-3	175B	**12,288** (published)
OpenAI GPT-4 / GPT-4o / GPT-5	Not public	❌ Unknown
Meta Llama 2	7B	4096
Llama 2	13B	5120
Llama 2	70B	8192
Llama 3 / 3.1	8B	4096
Llama 3 / 3.1	70B	8192
Google Gemma 3	1B	1024
Gemma 3	4B	2048
Gemma 3	12B	3072
Gemma 3	27B	5376
Mistral AI Mistral 7B	7B	4096
Mistral Nemo	12B	5120
Mistral Large	123B	Not officially published
Alibaba Cloud Qwen 2.5	7B	3584
Qwen 2.5	14B	5120
Qwen 2.5	32B	5120
Qwen 2.5	72B	8192
DeepSeek DeepSeek-V3 / R1	671B MoE	**7168** (Hugging Face)
Microsoft Phi-3 Mini	3.8B	3072
Phi-3 Medium	14B	5120
Phi-4	14B	~5120

Generative AI: Concepts and foundations: NLP

What is mean by number of parameters in GenAI?

Why parameters matter

Model capacity

- More parameters → can model **complex relationships**
- Fewer parameters → limited expressiveness

Knowledge storage

Parameters **store learned knowledge**:

- Grammar
- Facts
- Patterns
- Reasoning heuristics

LLMs do not "look up" answers — knowledge is embedded in parameters.

Why billions of parameters?

More parameters mean the model can represent more complex functions and patterns. Rough intuition:

Parameters	Capability
1 thousand	Simple regression
1 million	Image classifier
100 million	Small language model
7 billion	Good reasoning
70 billion	Expert-level language tasks
500+ billion (often MoE)	Frontier models

Analogy	Meaning
Parameters = knobs	Model adjusts knobs to fit data
Training = tuning knobs	Learning from examples
Inference = using tuned knobs	Generating output

Model	Approx. Parameters	What it implies
Small ML model	Thousands	Simple patterns
BERT-base	~110 million	Good language understanding
GPT-2	~1.5 billion	Fluent text generation
GPT-3	~175 billion	Reasoning + creativity
Modern LLMs	10B–1T+	Complex reasoning, multimodal

https://www.youtube.com/watch?v=bCz4OMemCcA

Generative AI: Concepts and foundations

Tokenization: the currency of LLMs

Aspect	Description
Concept	Computers process numbers, not text
Tokens	Text is chopped into chunks called "tokens"
Token vs Word	A token is not equal to a word (includes spaces, partial words)
Rule of Thumb	1,000 tokens ≈ 750 words
Impact	Determines API costs and model performance

Embeddings: The Map of Meaning

Aspect	Description
Concept	Tokens are converted into vector lists of numbers
Semantic Space	Words with similar contexts are mapped close together
Example	King − Man + Woman = Queen
Application	Powers semantic search and RAG systems

The Transformer & Attention Mechanism

Aspect	Description
Parallel Processing	Reads the whole sequence at once (unlike older AI models)
Self-Attention	Calculates relationships between every word
Example	Resolves what "it" refers to in a complex sentence
Result	Handles nuance, sarcasm, and complex instructions

Generative AI: Concepts and foundations

The Conceptual Shift

Aspect	Discriminative AI (Traditional)	Generative AI (New Paradigm)
Focus	Decision boundaries	Distribution mapping
Primary Task	"Is this a cat?" (Yes / No)	"Create a new image of a cat."

The Engine: Next Token Prediction

Aspect	Description
Core Mechanism	Giant autocomplete engine
Process	Calculates statistical probability of the next chunk
No "Knowledge"	Does not know facts, only probabilities
Non-Deterministic	Produces different output each time

Generative AI: Concepts and foundations

Traditional AI/ML Vs. Foundation Models

Aspect	**Traditional ML**	**Foundation Models**
Training data	Task-specific	Massive, diverse
Tasks	One task	Many tasks
Feature engineering	Manual	Learned automatically
Adaptation	Retrain	Prompt / fine-tune
Cost	Low training	High training, low reuse
Reasoning	Limited	Emergent

Feature	Traditional AI (Predictive)	Generative AI
Primary Function	Classification, Prediction, Recommendation.	Creation of new content (text, code, images, etc.).
Key Question	What is likely to happen? (e.g., Will this customer churn?)	What can we create? (e.g., Draft a personalized marketing email.)
Business Value	Efficiency, Risk Reduction, Optimization.	Innovation, Content Scale, Productivity Boost.
Key Models	Regression, Clustering, Decision Trees.	LLMs (GPT, etc.), Diffusion Models, GANs.

- Phase 1: Pre-Training (The Foundation)
  - Learns world representation.
  - Cost: Millions of dollars.
- Phase 2: Fine-Tuning (The Specialization)
  - Learns specific task behavior (e.g., Coding, Medical).
  - Cost: Low.

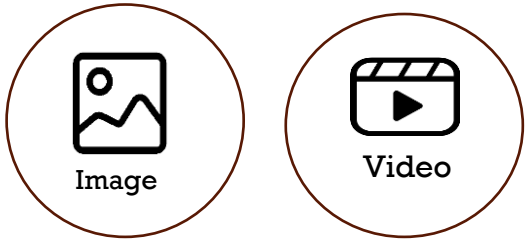
Generative AI: Concepts and foundations

Capabilities

GenAI Pattern	Software Engineering	Marketing & Content	Customer Operations	Legal & Internal Ops
1. Generation (Creating new content)	• Writing boilerplate code • Generating Unit Tests • Documentation drafting	• Blog post drafting • Social media variations • Ad copy generation	• Drafting email responses • Creating personalized scripts	• Drafting policy updates • Generating standard contracts
2. Summarization (Compressing info)	• Summarizing PRs (Pull Requests) • Release note generation	• Summarizing market reports • Meeting recap & action items	• Summarizing ticket history • Call log analysis	• Summarizing case law • Vendor contract summaries
3. Extraction (Unstructured ->Structured)	• Converting code to diagrams • Parsing logs to JSON	• Sentiment analysis tagging • Extracting keywords for SEO	• Extracting intent from emails • Categorizing feedback	• Invoice data extraction (PDF to CSV) • Compliance checking
4. Q&A / Search (Retrieval + Reasoning)	• "Talk to your Codebase" • StackOverflow-style bot	• Competitive intel bot • Product feature lookup	• RAG-based Support Bot • Agent assistant (Co-pilot)	• HR Policy Q&A Bot • Internal knowledge search

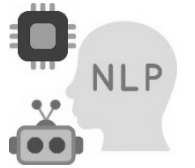
Generative AI: Concepts and foundations : Multi-modal capabilities

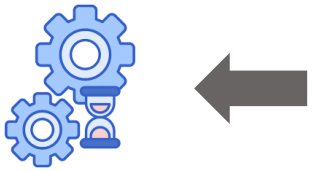
Multi-modal capabilities



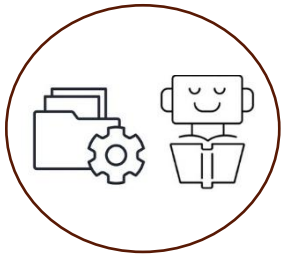
Human Interaction

Image Processing





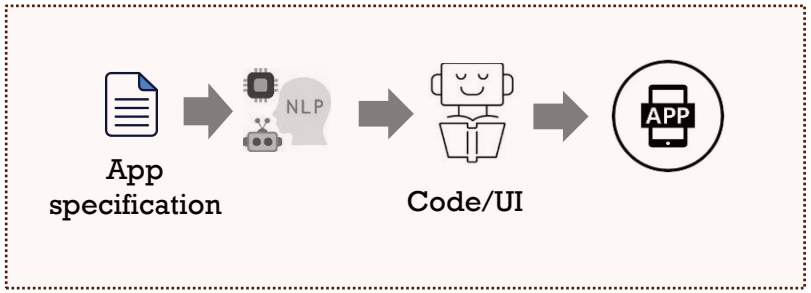
Automation of all kinds including devices, IOTs

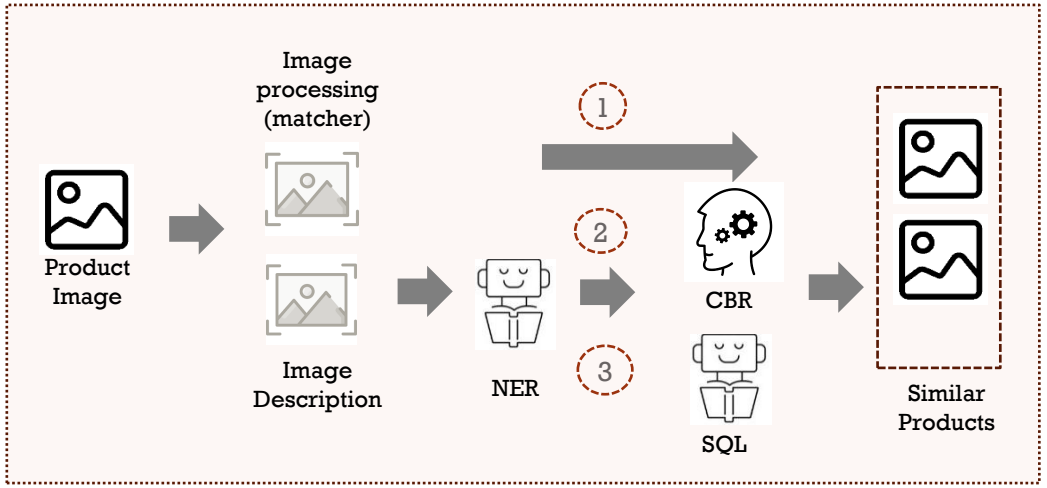


Technical Language | Standard Machine Processible

Seamless integration and understanding (semantics behind) of text, audio, images with technical (machines can understand) and human language. Conversion from one format to another one with ease.

Using effective combinations of these modes. Possible to address large number of use cases/applications





1. Computation heavy (training)
2. Lazy learning (semantics/personalization)
3. Less heavy. Simple search

NER: Named entity recognition
CBR: Case based reasoning

Generative AI: Concepts and foundations : Multi-modal capabilities

Multi-modal capabilities

Describe attributes and get in json format

LLM

```
{
  "brand": "Skechers",
  "category": "Hiking Shoes/Trail Shoes",
  "primary_color": "Olive/Green",
  "accent_color": "Black",
  "water_repellent": true,
  "insole": "Goga Mat",
  "outsole": "Goodyear Rubber",
  "design_series": "D-Lux Trekker",
  "closure_type": "Lace-Up",
  "comfort_technology": "D-Lux Comfort",
  "typical_uses": ["Outdoor Activities", "Casual Wear"]
}
```

JSON

Json-SQL

DB Search

Similar products

Input text

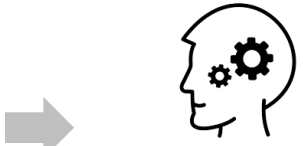
"hey I am looking for sketchers brown comfortable running shoes with lace and high white outsole within price range of INR 5000 to 10000"

Extract named entities in json format

LLM

```
{
  "brand": "Skechers",
  "color": "Brown",
  "category": "Running Shoes",
  "features": {
    "comfort": true,
    "closure_type": "Lace-Up",
    "outsole": "High",
    "outsole_color": "White"
  },
  "price_range": {
    "min": 5000,
    "max": 10000,
    "currency": "INR"
  }
}
```

JSON



CBR Semantic Search

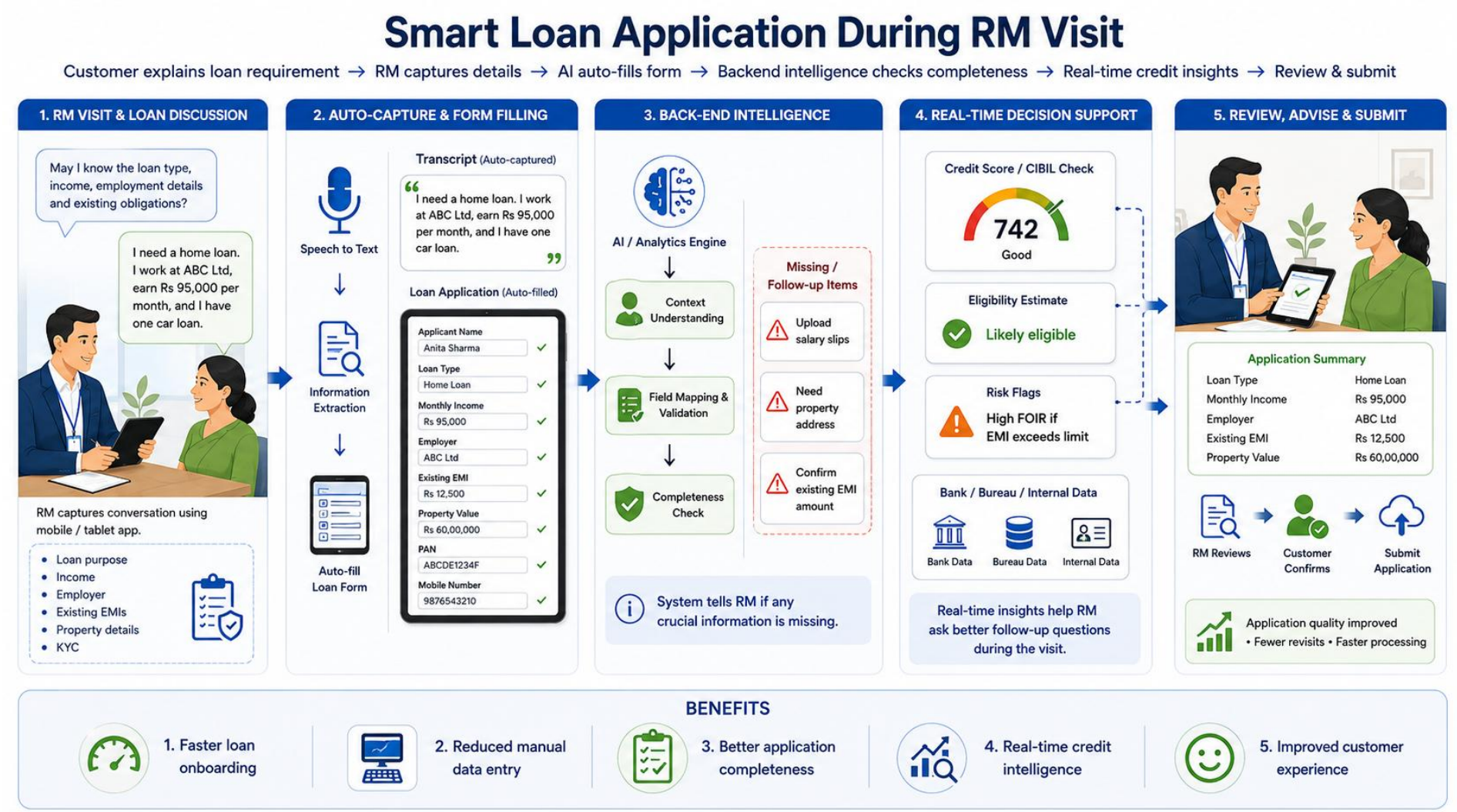
DB Search

Partial inputs can be given e.g. extract named entities from: "hey I am looking for sketchers brown comfortable running shoes with lace and high white outsole. Missing inputs would be asked during conversation

Better spend time on creating metadata about complex machine, design diagrams and use that metadata to search relevant images based on requirements. This won't need to burn GPUs for image recognition.

Generative AI: Concepts and foundations : Multi-modal capabilities

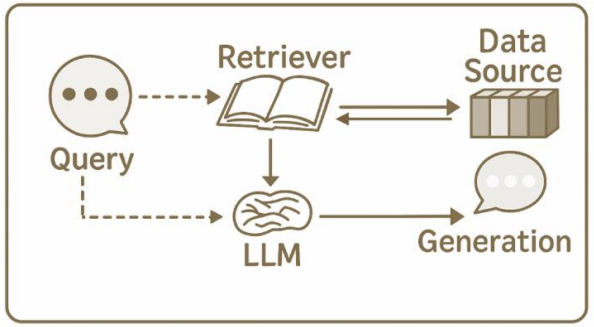
Multi-modal capabilities



Generative AI: Concepts and foundations : Issues and mitigation

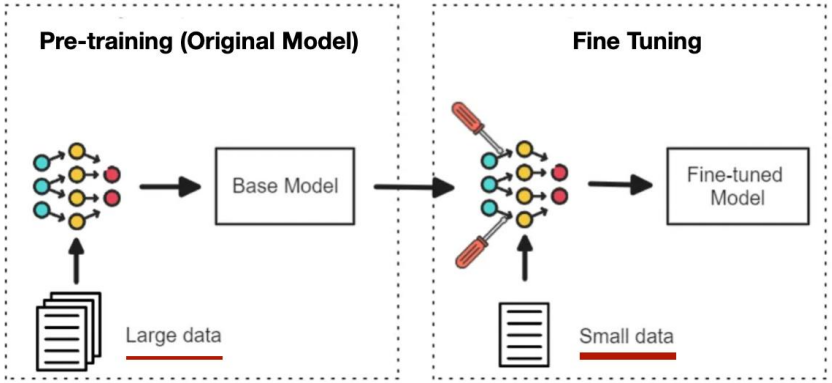
Limitation: The Hallucination Problem

Hallucination	
Issue	Model prioritizes plausibility over truth
Cause	Probabilistic guessing when facts are missing
Risk	Fabricated citations, case law, or history
Mitigation	Must use grounding sources



Solution: RAG

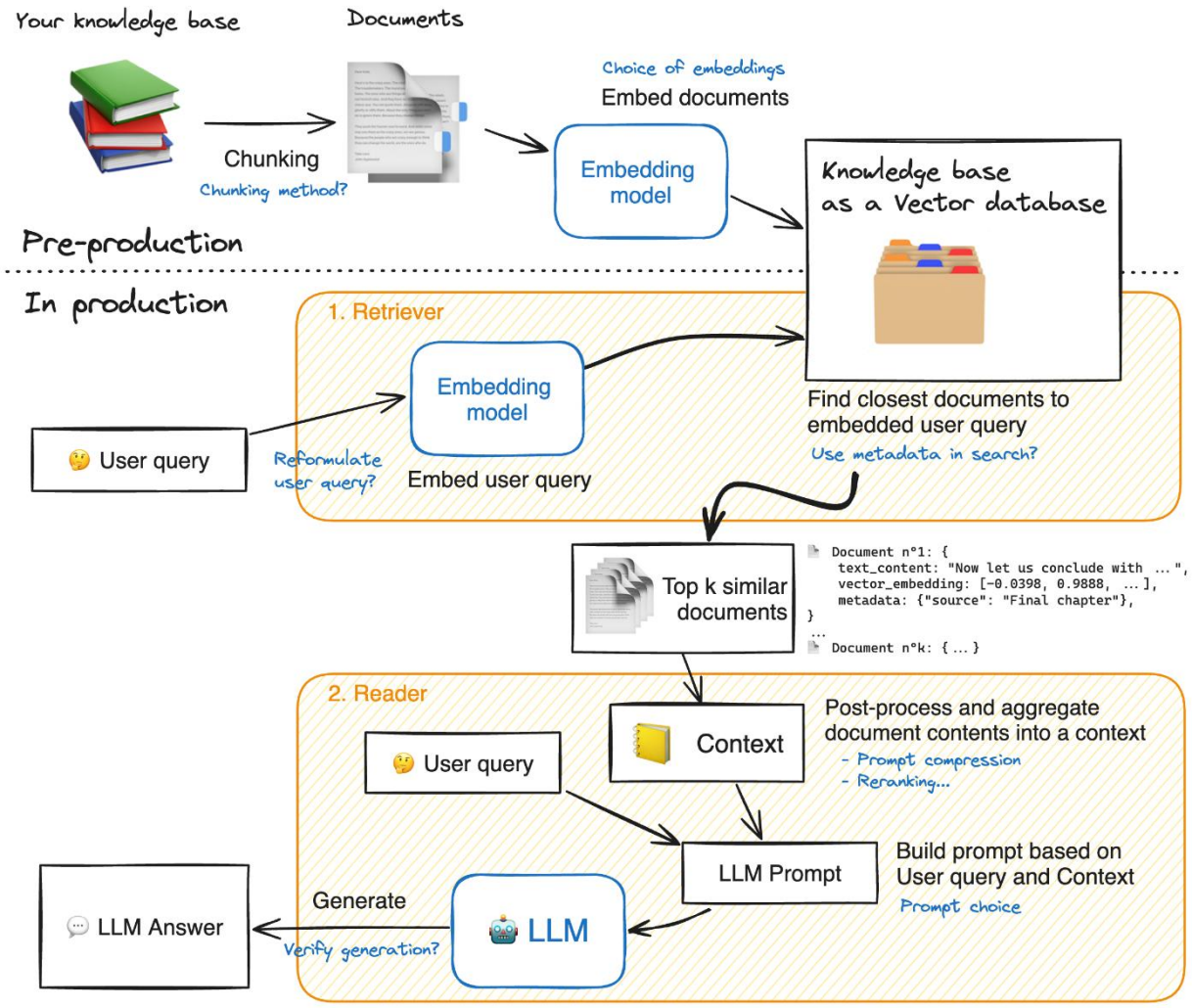
Context Window & Bias	
Context Window	The model's short-term memory
Constraint	Forgets start of conversation if too long
Frozen Knowledge	Does not know news from yesterday
IP Risk	Copyright status of generated content is unclear

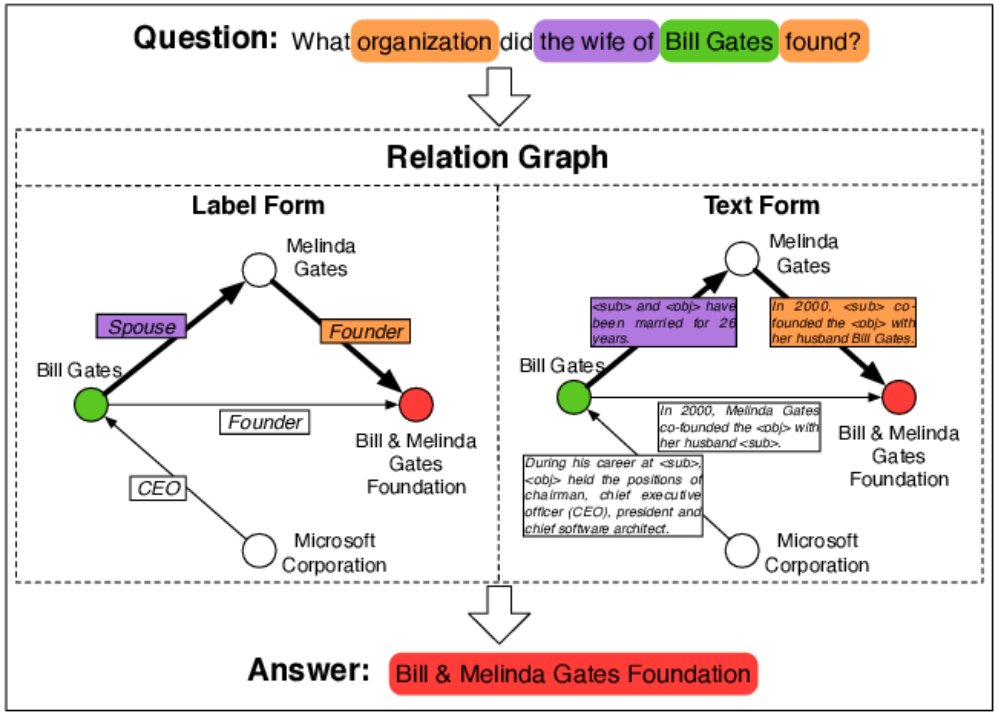


Solution: Fine tuning

Generative AI: Concepts and foundations : Issues and mitigation

Understanding RAG





Vector databases mostly deal with text data and not meant for numeric computational capabilities and searching capabilities (like search within data tables, decision tables, range tables etc.). Suggest not to include numeric (especially which are domain sensitive e.g. customer age in insurance, finance, health kind of domains) parameters. Use technologies like CBR for semantic retrieval.

Do not unnecessarily dump already structured data (like already stored in DBs unless lot of semantic and text data is involved) for the sake of it. Putting data in vector database limits search that is otherwise possible using complex SQL queries. Gen AI tools are good in converting user queries into SQL.

Context engineering can be challenging if lot of data resides in vector databases.

RAG System
An introduction to RAG and simple/ complex RAG

Fine tuning v/s RAG

Feature	Fine-Tuning	RAG
Primary Goal	Teaching the model a new behavior or style (e.g., speak like a doctor).	Giving the model new facts or knowledge (e.g., the Q3 sales report).
Data Freshness	Static (requires retraining to update).	Real-time (just update the document database).
Hallucination	Reduced, but still possible.	Drastically reduced (grounded in retrieved data).
Cost	High (training costs).	Low (retrieval infrastructure).

RAG vs Fine Tuning vs Prompt Engineering

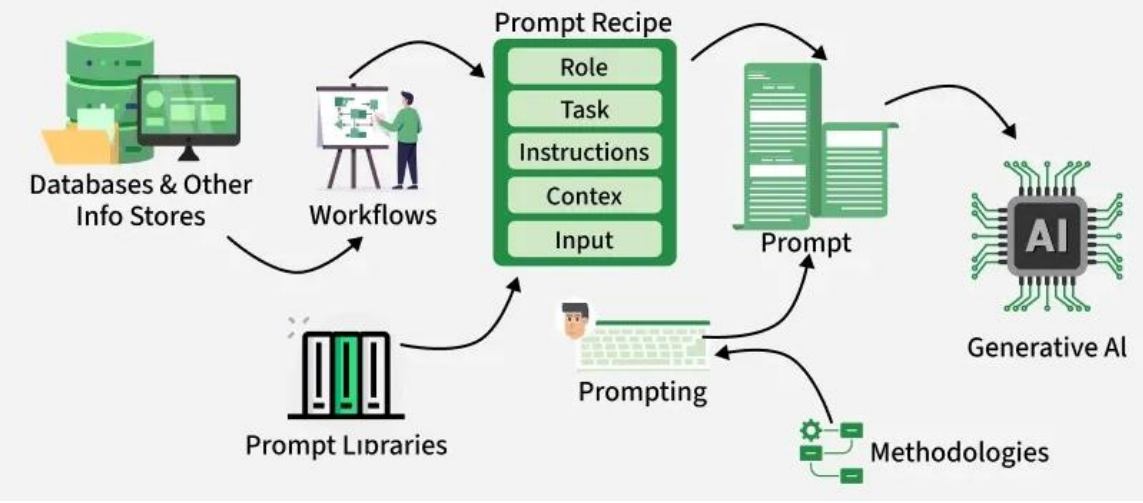
Limitations and issues with RAG

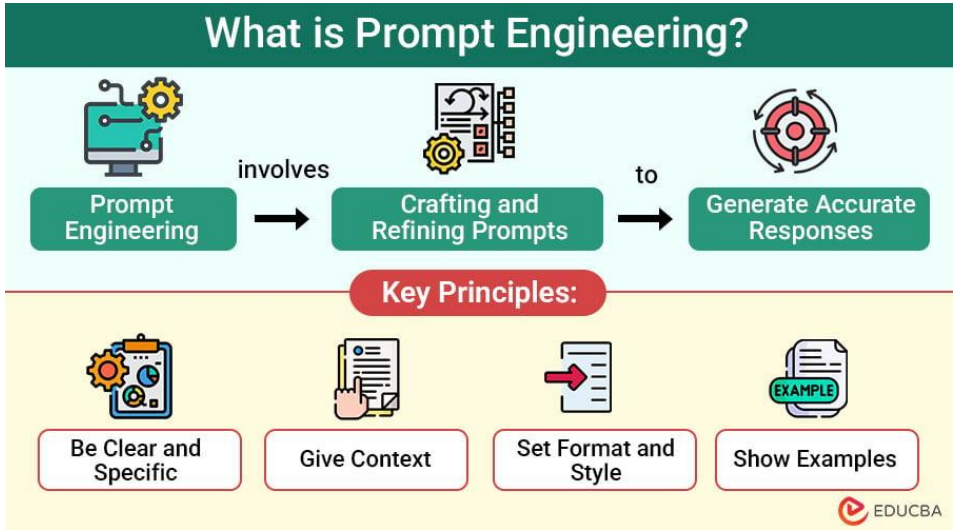
Problem Area	**Mitigation Strategy**
Retrieval errors	Hybrid search
Chunking issues	Adaptive chunking using headings/sections
Embedding gaps	Domain-specific embedding models
Hallucinations	Citation-enforced prompting, answer grounding
Latency & cost	Caching, top-K tuning, context pruning
Freshness	Incremental indexing, TTLs (Time to live), streaming ingestion
Debugging	Separate retrieval & generation evaluation
Security	Doc-level ACLs (access level lists), injection filtering
Scalability	Sharded vector DBs, tenant isolation
Reasoning	Tool calling, rule engines, graphs

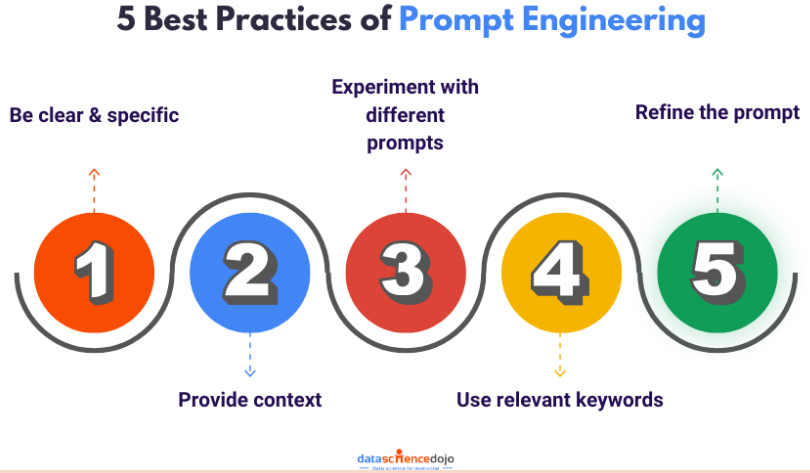
Generative AI: Concepts and foundations

Prompt Engineering

Prompt engineering is the practice of designing and refining input instructions (prompts) to guide generative AI models, like large language models (LLMs), to produce accurate, relevant, and desired outputs, essentially "programming" the AI in natural language to achieve specific goals, from writing code to creating content. It involves understanding model capabilities, providing context, specificity, and structure, often through techniques like few-shot or chain-of-thought prompting, to bridge the gap between human intent and machine understanding







[What-is-an-ai-prompt-engineering]

[Prompt-engineering-best-practices]

[Prompt-engineering]

[Prompt Engineering Types]

Prompts can change based on user is tech user, general user or domain specialist and business user. Prompts used (**context engineering!**) to automate LLM tasks (like in AI agents) using LLMs (mostly driven by APIs) are structured and dynamic; and need to be modelled differently especially with formatted input and output.

Generative AI: Concepts and foundations: Prompts for what?

Who is cognitive worker!

Content Generation

- Create contents (text, images, videos, docx, PPTs, etc.)
- Generate technical contents (code, database schema etc.)

Multi-modal tasks

- Convert content from one mode to another e.g. 1> converting audio to transcripts, 2> extracting text from image (e.g. OCR), 3> explaining pictures based on the context

Language translation

- Translation from one language to another e.g English to Hindi

Conversion of data

- Converting from one format to another format (this can be simple plain or semantic conversion) e.g. 1> excel horizontal format (tedious to read) to vertical format word doc, 2> CSV to JSON, 3> CSV to database queries
- Convert from unstructured to technical formats like Json, XML, Rule, Code etc. e.g.
  Image to meta-data in attribute value pair,
  NLP text into JSON,
  HR leave rules into rules,
  NLP text into SQL (based on schema provided),
  NLP text into named entity pairs…

Template based extraction

- Extracts data from predefined templates e.g. PAN card to data
- Reading forms/printed outputs

Summarization and insights

- Get insights from documents (excel, docs, pdfs etc.)
- Showing contents using visualization

Research and exploration

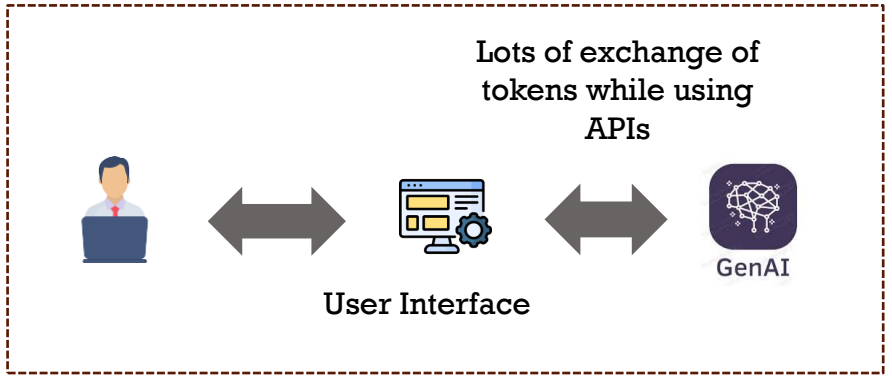
- Review kind of work (understand what has happened, what is happening, who has done/said what etc.)
- Market research on specific product, getting validations done and understanding current status (e.g. where your product fits, what are the competing products, how their features differ from your product's features?)
- Exploring how you can move from where are you right now to the target (e.g. 1> how you can reach to niche product X1 from your existing product X, 2> how you can enhance your skills to specific job)

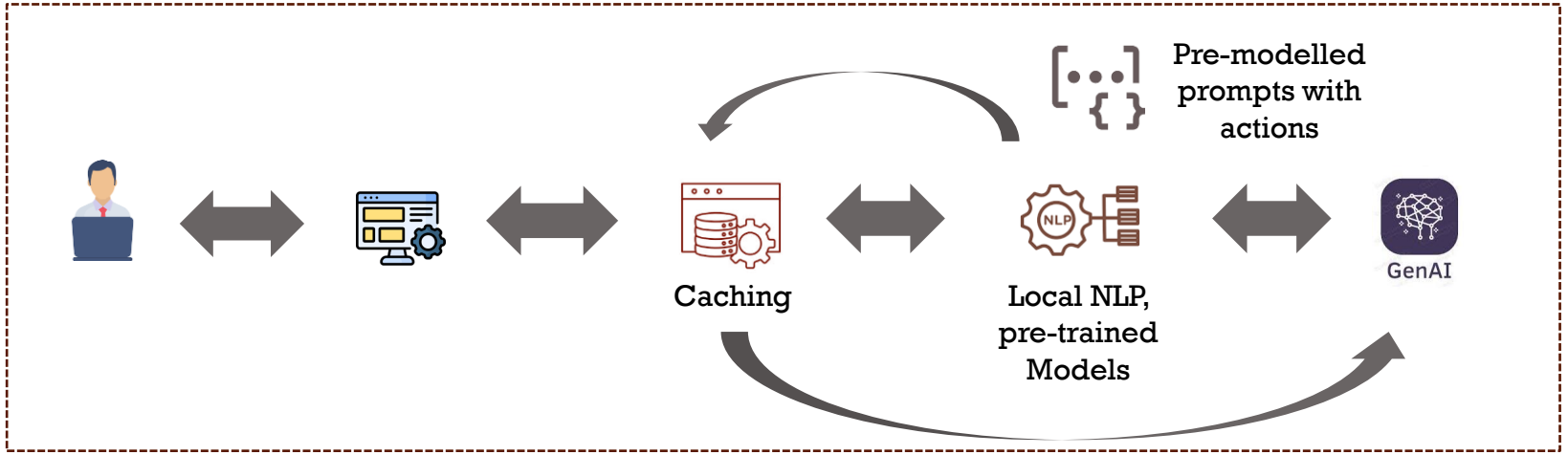
To create prompt itself

- If you don't know a specific prompt for your task you can ask GenAI to create on for you and you can go on refining it

Generative AI: Concepts and foundations

Optimizing GenAI usage (every API call and token gets counted!)

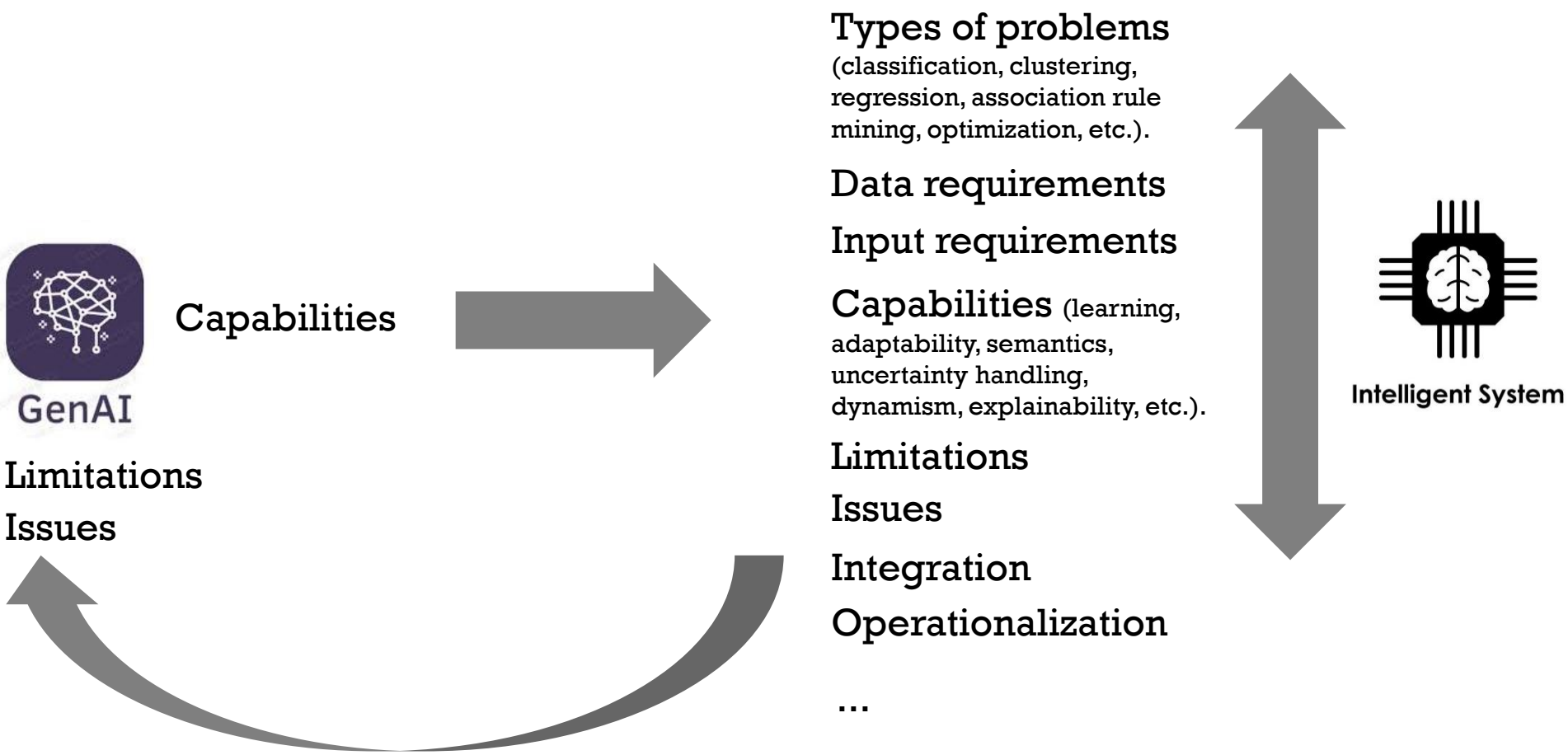




Optimal use of AI needs lot of thinking especially from optimal usage of compute, main memory, data storage and access, network bandwidth, exchange of data, paid APIs, etc. Caching and local storage e.g. frequently used data can be kept in cache and local storage, local processing to great extent instead of using APIs etc.

Data scientists and ML developers may not understand this fully. Better to have core and smart CS/IT person having good understanding of software engineering concepts to design and implement AI or AI backed systems. Most of the time, the schema/meta-data of data is enough and data can be replaced with placeholders e.g. NLP to SQL, only DB schema is enough to get queries written by Gen AI.

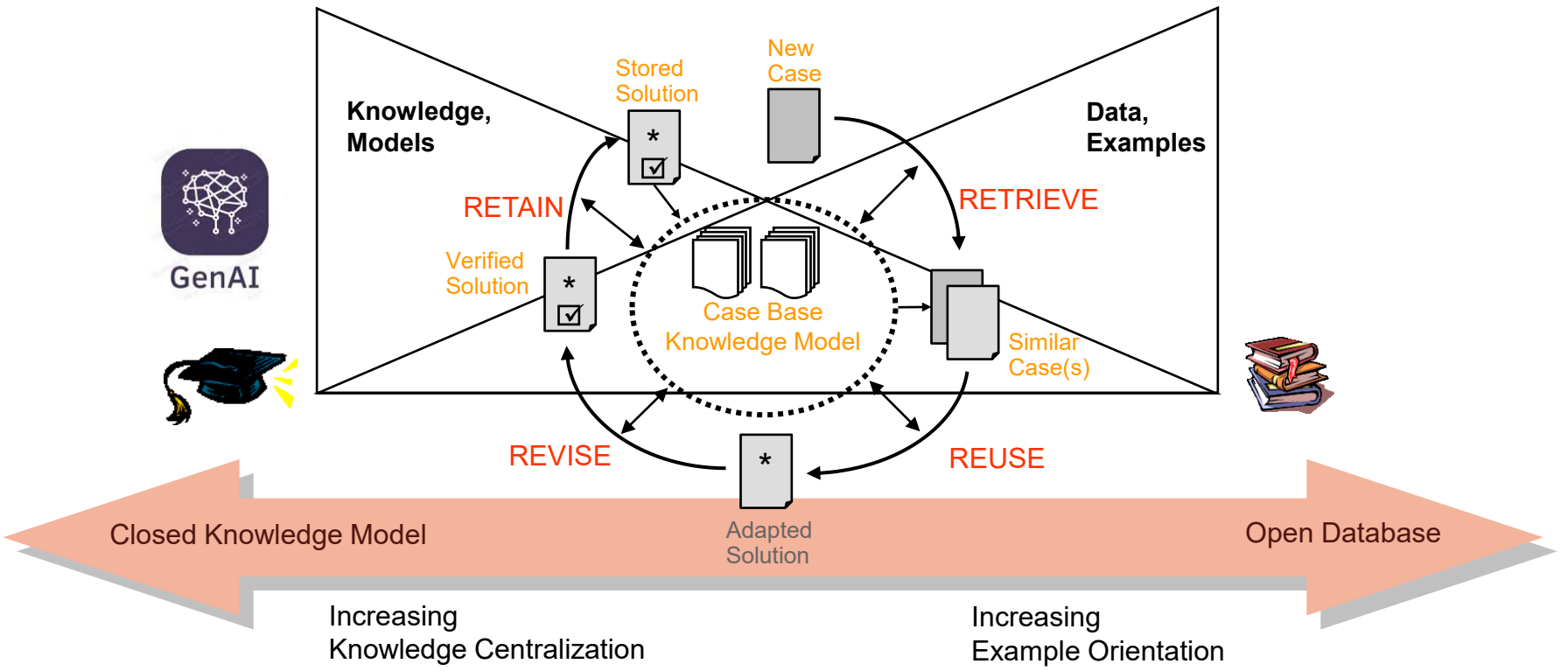
Integrating Gen AI and Traditional AI/ML Systems



GenAI can take care of issues and limitations of intelligent systems. It can help to contextualize, integrate and operationalize these systems making it possible to leverage classical AI technologies in the current context. Intelligent systems can reduce hallucination, enhance accuracy and explainability of knowledge/insights from LLM.

Images from different sources are acknowledged

Integrating Gen AI and Traditional AI/ML systems



Expert Systems
(completely knowledge driven)

CBR Systems
(knowledge + data driven)

Databases
(completely data driven)

Integrating Gen AI and Traditional AI/ML Systems

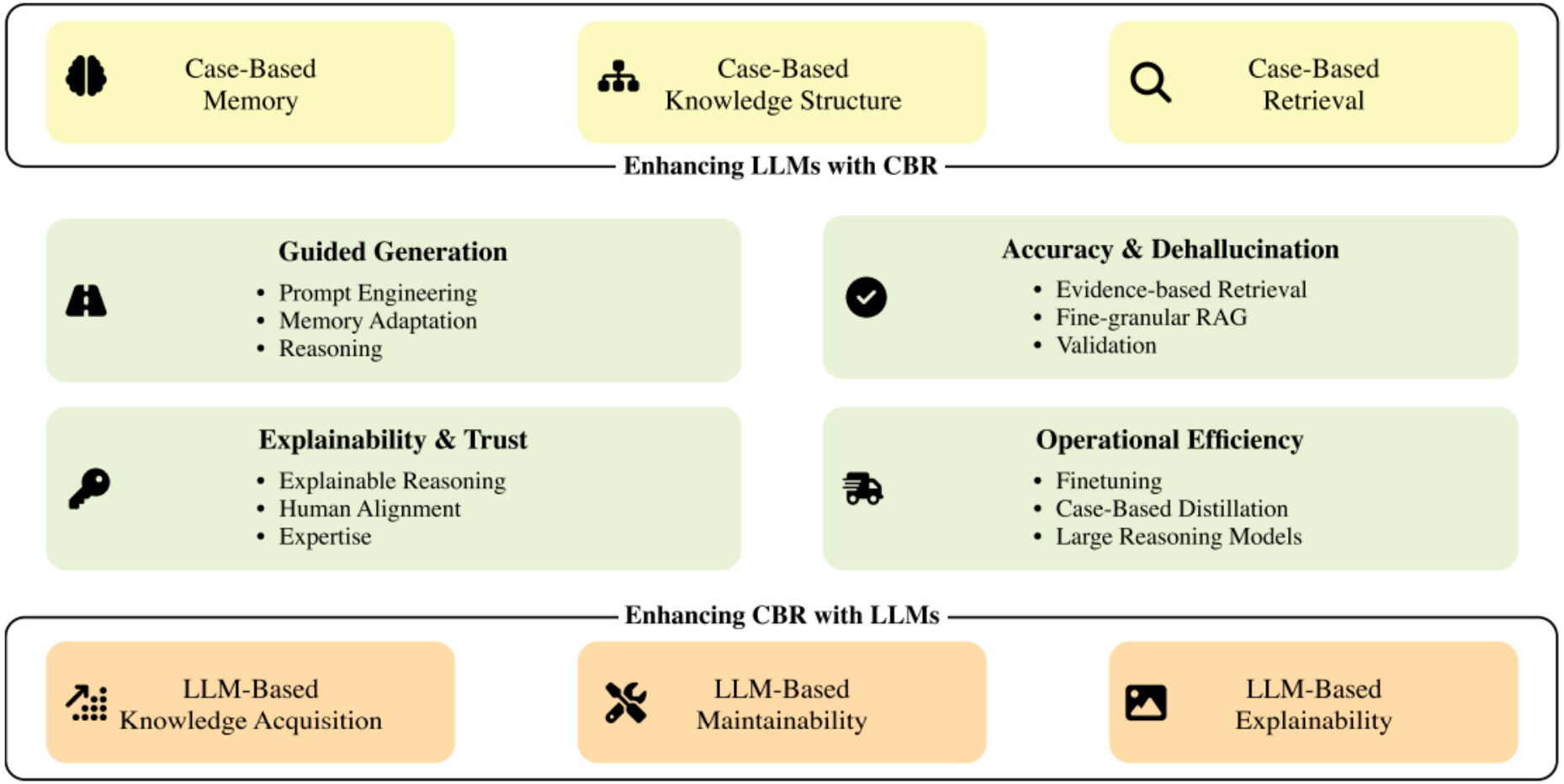


FIGURE 1 Overview of Open Challenges for CBR-LLM Integration

CBR Meets Large Language Models: A Research Manifesto For Open Challenges and Research Directions

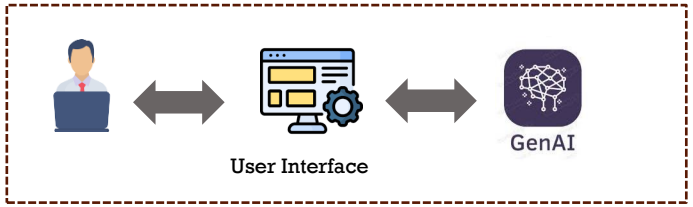
Integrating Gen AI and Traditional AI/ML Systems

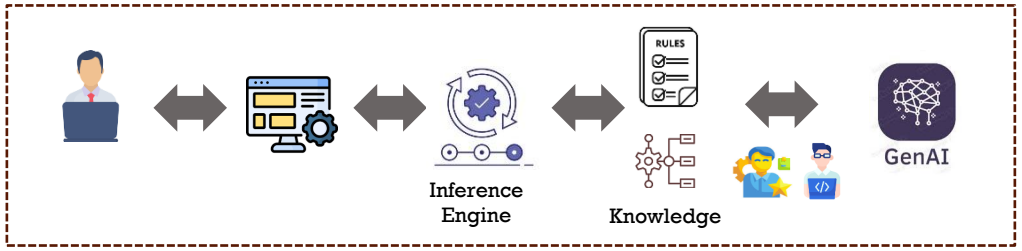
LLM-ES Integration:

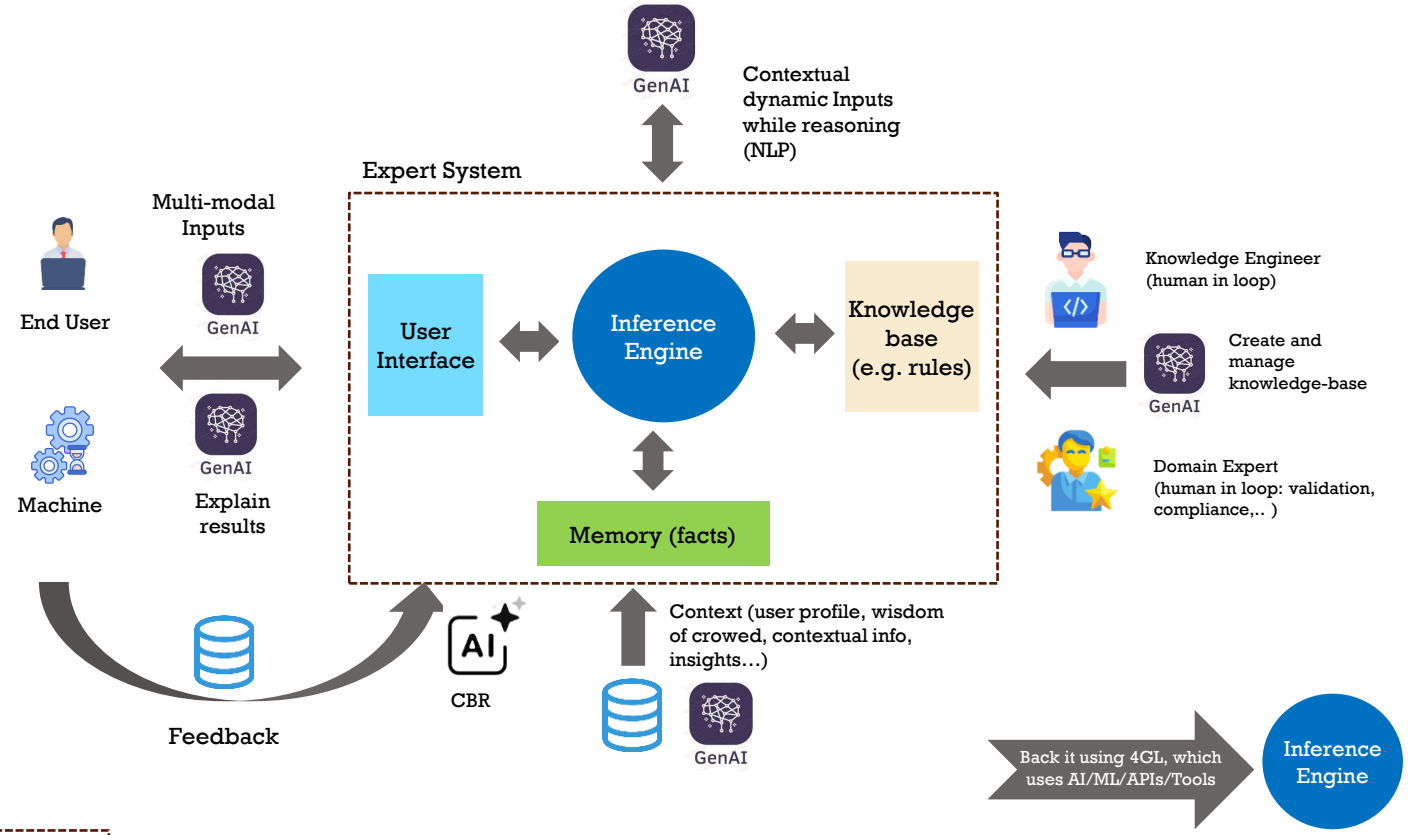
Conclusion: Coexistence Over Competition

While **generative AI** dazzles with creativity, expert systems stand firm as the backbone of structured problem-solving. Their enduring value lies in their **predictability, transparency, and precision**—qualities that generative AI often cannot guarantee.

The future isn't about choosing one over the other. Instead, businesses can harness the strengths of both systems to create **powerful, hybrid solutions** that combine innovation with reliability. Expert systems may not dominate headlines anymore, but their legacy—and utility—is far from fading.





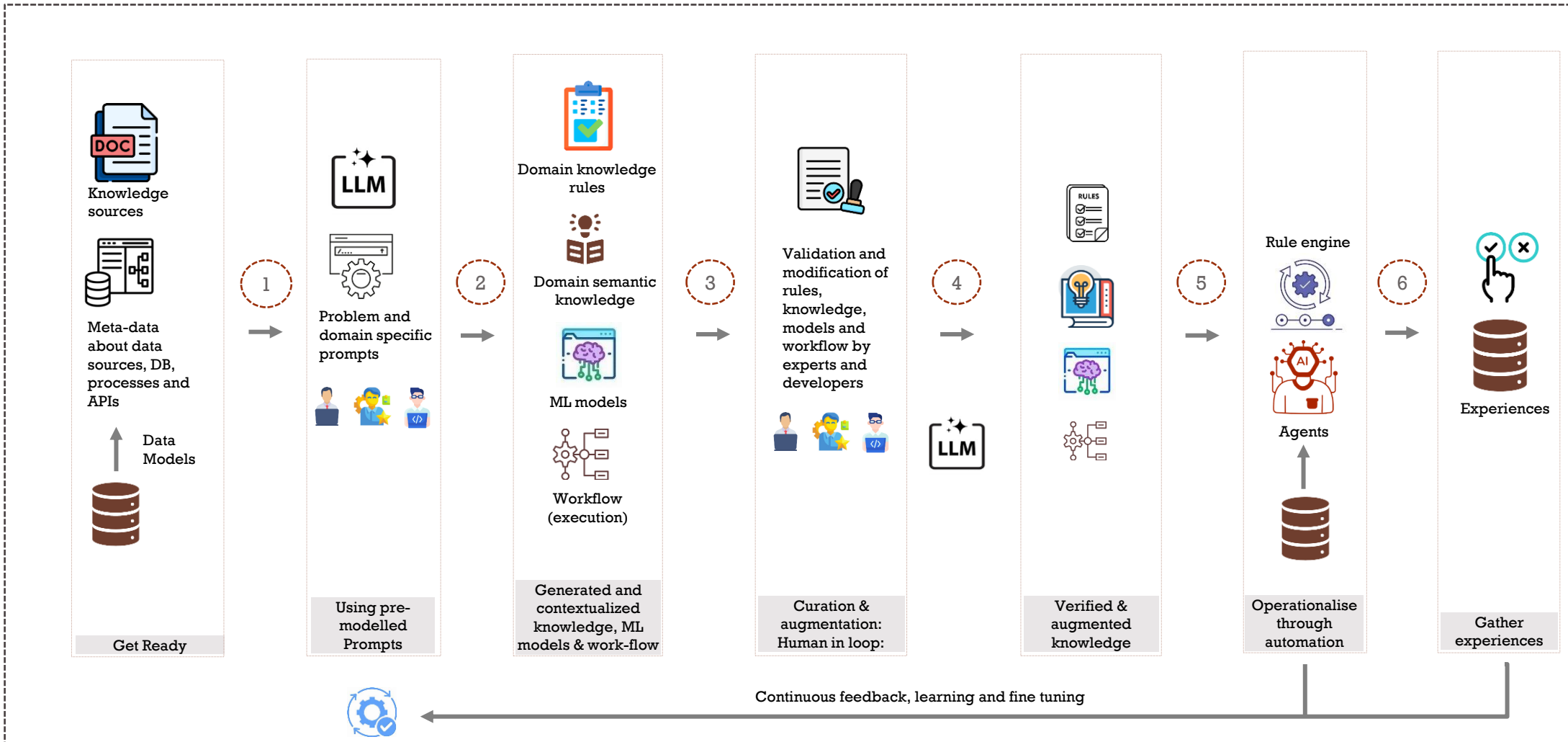


How Gen AI can support ES

Why Expert Systems Still Matter in the Age of Generative AI

Leveraging Classical AI with Generative AI, Exploring Possibilities: Special focus on BFSI space

Integrating Gen AI and Traditional AI/ML systems



Integrating Gen AI and Traditional AI/ML systems

Automating PDF (rule-book)

ES: Possible steps

- ❑ Get your PDF(s) to be automated
- ❑ Keep your database schema and specifications of APIs/connectors ready which you need to connect to operational systems, databases etc.
- ❑ Choose expert system software which supports many functions including database functions, consuming APIs, integrating tools, adding ad-hoc functionality etc.
- ❑ Define possible goals the user would think of, e.g. for leave management system they can be avail leave, leave entitlement, enquiry about leaves availed, pending etc. for expert system
- ❑ Optionally build local NLP capabilities using sentence transformer to match the predefined goals. NLP can also be used to derive the facts based on what user is looking for using Gen AI with minimal inputs.
- ❑ Create prompts with rule-format and other knowledge components such as taxonomies supported by expert system software and possible goals etc.
- ❑ Build rules using Gen AI by giving the prompts. Optionally create explanation details with GenAI based on the goals specified.
- ❑ Develop expert system using rules built and test the system
- ❑ Validate rules: 1> sample runs and check the results by experts or match with rule-book 2> use Gen AI to test outcomes with predefined scenarios, 3> inspect rules with help of experts
- ❑ Put system into production without Gen AI or limited Gen AI for input/output
- ❑ Collect feedback from users

- ❑ Keep updating whenever required (e.g. change in rule, feedback, missing goals etc.)

Key takeaways

Neural networks can address complex set of problems; generative AI is a complete paradigm shift and disruptive.

Deep learning
Understand where deep learning becomes relevant. Neural networks are core of deep learning and capable of solving problems involving NLP, vision, audio processing etc.

Generative AI
Understanding foundations of generative AI is important to know inherent its problem-solving capabilities. Prompt and context engineering is essential to leverage capabilities. It can increase productivity many-fold if done correctly.

Generative AI is disruptive technology. Needs AI data readiness, right expertise and strong governance to leverage it.

Addressing issues and optimizing usage of Generative AI
Role of RAG, Fine-tuning etc. in reducing hallucination and make generative AI more context and domain specific. How caching, local NLP using sentence transformers can reduce tokens at run-time.

Integrating Classical AI and Generative AI
Classical AI models and solves many kinds of problems in more deterministic way, Don't have overheads of tokens at run-time like Generative AI APIs.
Organizations can leverage classical AI and generative AI together if they understand them well.