Artificial Intelligence, Machine Learning, and Deep Learning

IIT ROPAR Minor In AI

21 March, 2025

Contents

1 Introduction to AI, Machine Learning, and Deep Learning

1.1 AI: Mimicking Human Intelligence

Artificial Intelligence (AI) refers to systems designed to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation.

Historical Context

The term "Artificial Intelligence" was coined by John McCarthy in 1956 at the Dartmouth Conference, which is considered the founding event of AI as a field. Early AI systems were predominantly rule-based and focused on symbolic reasoning.

Case Study: IBM's Deep Blue

The 1997 chess match between IBM's Deep Blue and world champion Garry Kasparov represented an early milestone in AI. Deep Blue used a combination of brute-force computation and sophisticated evaluation functions to defeat Kasparov, demonstrating how machines could outperform humans in specific domains through different approaches than human cognition.

$$\text{AI System} = \begin{cases} \text{Task Performance} \\ \text{Learning Capability} \\ \text{Adaptability} \\ \text{Reasoning Mechanisms} \end{cases} \tag{1}$$

1.2 Machine Learning: Learning from Data without Explicit Coding

Machine Learning (ML) is a subset of AI that focuses on building systems that can learn from and make decisions based on data, without being explicitly programmed for specific tasks.

$$f : X \to Y \tag{2}$$

Where X represents input data and Y represents output predictions. The function f is learned from training data rather than being explicitly defined.

Key ML Paradigms:

- **Supervised Learning:** Training on labeled data
- **Unsupervised Learning:** Finding patterns in unlabeled data

- **Reinforcement Learning:** Learning through interaction with an environment

Case Study: Netflix Recommendation System

Netflix employs machine learning algorithms to analyze user viewing history, ratings, and preferences to recommend content. This system processes billions of data points to learn patterns that predict which shows a user might enjoy, demonstrating how ML can create personalized experiences at scale. The recommendation system combines collaborative filtering (comparing user behavior with similar users) and content-based methods (analyzing show attributes).

1.3 Deep Learning: Inspired by Human Brain, Uses Neural Networks

Deep Learning is a subset of machine learning that uses neural networks with multiple layers (hence "deep") to progressively extract higher-level features from raw input.

$$y = \sigma(w \cdot x + b) \tag{3}$$

Where σ is an activation function, w represents weights, x is the input, and b is a bias term.

Case Study: AlphaFold

DeepMind's AlphaFold represents a breakthrough application of deep learning in protein structure prediction. Prior to AlphaFold, determining protein structures was an enormously time-consuming laboratory process. AlphaFold uses deep neural networks trained on known protein structures to predict the three-dimensional structure of proteins from their amino acid sequences with unprecedented accuracy, revolutionizing molecular biology and drug discovery.

2 Phases of AI: Rule-based, Predictive, Generative, Agentic

2.1 Rule-based AI (1950s-1990s)

Rule-based AI systems operate using explicitly programmed rules in the form of if-then statements.

Listing 1: Example of Rule-Based AI in Prolog

```
% Facts
parent(john, mary).
parent(john, tom).
parent(mary, ann).

% Rules
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
```

Case Study: MYCIN

MYCIN was an early expert system developed at Stanford University in the 1970s to diagnose infectious blood diseases and recommend antibiotics. It contained approximately 600 rules that encoded the knowledge of infectious disease experts. When tested, MYCIN performed at a level comparable to specialists, demonstrating how explicit rules could capture expert knowledge.

2.2 Predictive AI (1990s-2010s)

Predictive AI uses statistical methods and machine learning to make predictions based on patterns in data.

Case Study: Credit Scoring Models

Financial institutions use predictive AI to assess credit risk. These systems analyze factors such as payment history, debt levels, and income to predict the likelihood of loan repayment. Modern credit scoring systems use ensemble methods combining multiple models (decision trees, logistic regression, etc.) to improve prediction accuracy. These models have transformed lending by enabling more objective, data-driven decisions.

2.3 Generative AI (2010s-Present)

Generative AI creates new content (text, images, audio, etc.) that resembles human-created content.

$$P(x_t|x_{<t}) = \text{softmax}(W \cdot h_t + b) \tag{4}$$

Where x_t is the next token to be generated, $x_{<t}$ represents previous tokens, and h_t is the hidden state.

Case Study: DALL-E

OpenAI's DALL-E demonstrates the capabilities of generative AI in visual domains. Given a text prompt like "an astronaut riding a horse in a photorealistic style," DALL-E can generate original images that integrate these concepts. This demonstrates how generative models can combine concepts in creative ways never explicitly shown during training, exhibiting a form of artificial creativity.

2.4 Agentic AI (Emerging)

Agentic AI systems can operate autonomously, make decisions, and take actions to achieve specified goals.

Agentic AI Framework

1. Perception: Understanding the environment
2. Planning: Determining action sequences
3. Execution: Implementing planned actions
4. Learning: Improving from experiences

Case Study: AutoGPT

AutoGPT represents an early example of agentic AI application. It combines large language models with the ability to use tools (web search, file operations, etc.) and maintain a memory of past actions. Given a high-level objective like "research the market for electric vehicles and write a report," AutoGPT can break this down into sub-tasks, execute them sequentially, and produce the desired output with minimal human intervention, demonstrating autonomous goal-directed behavior.

3 Deep Learning Basics

3.1 Inspiration from Human Brain Neurons

Artificial neural networks draw inspiration from the structure and function of biological neurons in the human brain.

$$\text{Biological Neuron} \rightarrow \text{Artificial Neuron} \tag{5}$$

$$\text{Dendrites} \rightarrow \text{Input Weights} \tag{6}$$

$$\text{Cell Body} \rightarrow \text{Summation \& Activation} \tag{7}$$

$$\text{Axon} \rightarrow \text{Output} \tag{8}$$

While artificial neurons are vast simplifications of biological neurons, they capture the essential computational elements: receiving weighted inputs, integrating them, and producing an output if the integrated signal exceeds a threshold.

3.2 Perceptrons and Multi-layer Neural Networks

The perceptron is the fundamental building block of neural networks.

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{9}$$

$$a = \sigma(z) \tag{10}$$

Where w_i are weights, x_i are inputs, b is bias, and σ is an activation function. Multi-layer networks stack these units to create more complex architectures:

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \tag{11}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]}) \tag{12}$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \tag{13}$$

$$\mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]}) \tag{14}$$

Case Study: XOR Problem

The XOR problem (exclusive OR) illustrates why multi-layer networks are necessary. A single perceptron cannot solve the XOR problem because it's not linearly separable. However, a neural network with at least one hidden layer can learn this function. This simple example demonstrates how adding layers enables networks to represent increasingly complex functions and decision boundaries.

3.3 Convolutional Neural Networks (CNN) for Image Processing

CNNs apply convolutional operations to extract spatial features from images.

$$(f * g)(x, y) = \sum_{m} \sum_{n} f(m, n) g(x - m, y - n) \tag{15}$$

Key Components:

- Convolutional layers: Extract features using learnable filters
- Pooling layers: Reduce dimensionality while preserving important information
- Fully connected layers: Final classification based on extracted features

Case Study: ResNet

Residual Networks (ResNet) addressed the problem of training very deep CNNs by introducing skip connections that allow gradients to flow more easily through the network. This innovation enabled the creation of networks with over 100 layers that could be effectively trained. ResNet dramatically improved image classification performance on the ImageNet dataset and became a foundational architecture for many computer vision applications.

3.4 Recurrent Neural Networks (RNN) for Sequential Data

RNNs process sequential data by maintaining a hidden state that captures information from previous timesteps.

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{16}$$
$$\mathbf{y}_t = \sigma(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y) \tag{17}$$

LSTM (Long Short-Term Memory) networks address the vanishing gradient problem in traditional RNNs:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{18}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{19}$$
$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \tag{20}$$
$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{C}}_t \tag{21}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{22}$$
$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{C}_t) \tag{23}$$

Case Study: Neural Machine Translation

Google's Neural Machine Translation (GNMT) system demonstrated the power of RNNs in sequence-to-sequence learning. Prior to the transformer architecture, GNMT used bidirectional LSTMs with attention mechanisms to translate between languages. The system showed significant improvements over phrase-based statistical methods, especially for grammatically complex language pairs like English-Japanese, by capturing long-range dependencies and context.

4 Transformers and Attention Mechanism

4.1 Google's "Attention is All You Need" Paper (2017)

The landmark paper by Vaswani et al. introduced the transformer architecture, which revolutionized natural language processing by eliminating recurrence and convolutions in favor of attention mechanisms.

Transformer Architecture

Key innovations:

- Self-attention mechanism
- Positional encoding
- Multi-head attention
- Feed-forward networks in each layer

4.2 Self-attention and Parallel Processing Capabilities

Self-attention computes relationships between all positions in a sequence:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{24}$$

Where Q (queries), K (keys), and V (values) are derived from the input sequence.

Multi-head attention computes attention multiple times in parallel:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{25}$$

Where each head performs attention with different linear projections.

Case Study: BERT

Google's Bidirectional Encoder Representations from Transformers (BERT) demonstrated the power of transformer architectures for language understanding. BERT is pre-trained using masked language modeling and next sentence prediction objectives on a large corpus of text. When fine-tuned on specific tasks, BERT achieved state-of-the-art results on a wide range of natural language understanding benchmarks. Its bidirectional attention mechanism allows it to consider context from both directions, improving performance on tasks like question answering and sentiment analysis.

Parallel Processing Advantage:

Unlike RNNs, which process sequences element by element, transformers process entire sequences in parallel:

$$\text{RNN Complexity} = O(n) \text{ sequential operations} \tag{26}$$

$$\text{Transformer Complexity} = O(1) \text{ sequential operations} \tag{27}$$

This parallelization enables efficient training on modern GPU hardware, allowing for much larger models.

5 Large Language Models (LLMs)

5.1 Training Process: Pre-training, Post-training, Reinforcement Learning

5.1.1 Pre-training

During pre-training, models learn general language patterns from vast amounts of text.

Pre-training Scale

Modern LLMs are trained on:

- Hundreds of billions to trillions of tokens
- Diverse sources: books, websites, code, research papers
- Months of computation on thousands of GPUs

5.1.2 Post-training (Fine-tuning)

After pre-training, models are adapted for specific capabilities:

- **Supervised Fine-tuning (SFT):** Using human-created demonstrations
- **Instruction Tuning:** Teaching models to follow user instructions

5.1.3 Reinforcement Learning from Human Feedback (RLHF)

RLHF aligns model outputs with human preferences:

$$L_{\text{RLHF}} = E_{x \sim D}[r_\phi(x, y) - \beta \log \frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)}] \tag{28}$$

Where r_ϕ is a learned reward model based on human preferences, and the second term is a KL-divergence penalty to prevent excessive deviation from the reference model.

Case Study: ChatGPT

OpenAI's ChatGPT illustrates the full LLM training pipeline. Starting with a GPT architecture pre-trained on a diverse text corpus, it underwent instruction tuning to follow user directions and RLHF to align with human preferences. This process transformed a general text prediction model into an assistant that could respond helpfully to user queries, follow instructions, and generate more useful, safe, and truthful responses. Its capabilities and limitations demonstrate both the potential and challenges of current LLM technology.

5.2 Applications and Limitations

5.2.1 Applications

- **Content Generation:** Writing, summarization, translation
- **Code Assistance:** Generating, explaining, and debugging code
- **Conversational AI:** Customer service, digital assistants
- **Information Extraction:** Analyzing documents, reports

5.2.2 Limitations

Hallucination: LLMs can generate plausible-sounding but factually incorrect information.

Example of Hallucination

When asked about obscure topics, LLMs may confidently generate fictional information, such as inventing non-existent research papers or creating false historical events.

Knowledge Cutoff: LLMs cannot know about events after their training data ends.

$$\text{Knowledge Access} = \begin{cases} \text{Comprehensive} & \text{for } t < t_{\text{cutoff}} \\ \text{None} & \text{for } t > t_{\text{cutoff}} \end{cases} \tag{29}$$

Context Length: LLMs have a finite window of text they can process at once.

$$\text{Maximum Context} = n \text{ tokens} \tag{30}$$

Where n has increased from about 2,048 in early models to 128,000+ in recent architectures.

Case Study: Mitigating Limitations in Claude

Anthropic's Claude demonstrates approaches to addressing LLM limitations. To reduce hallucinations, Claude was trained using constitutional AI methods that encourage the model to express uncertainty rather than confabulate when asked about topics outside its knowledge base. To overcome context limitations, Claude implements techniques for efficient context compression and retrieval, allowing it to process longer documents while maintaining coherent understanding.

6 Prompt Engineering

6.1 Types: Zero-shot, Few-shot, Chain of Thought

6.1.1 Zero-shot Learning

The model performs tasks without specific examples:

Listing 2: Zero-shot Prompt

```
Classify the following text as either positive or negative:
"The service at this restaurant was terrible and the food was cold."
```

6.1.2 Few-shot Learning

Providing examples helps the model understand the desired pattern:

Listing 3: Few-shot Prompt

```
Classify reviews as positive or negative:

Review: "Amazing food and excellent service!"
Sentiment: Positive

Review: "Waited an hour and the food was bland."
Sentiment: Negative

Review: "The ambiance was nice but overpriced for what you get."
Sentiment:
```

6.1.3 Chain of Thought (CoT)

Encouraging step-by-step reasoning improves performance on complex tasks:

Listing 4: Chain of Thought Prompt

```
Question: If a store has 10 apples and sells 3 to customer A and 4 to customer B

Let's think through this step by step:
1. The store starts with 10 apples.
2. It sells 3 apples to customer A, leaving 10 - 3 = 7 apples.
3. It sells 4 apples to customer B, leaving 7 - 4 = 3 apples.
4. It buys 5 more apples, giving it 3 + 5 = 8 apples total.

Therefore, the store has 8 apples now.
```

Case Study: GSM8K Math Problems

Research on the GSM8K benchmark (grade school math problems) demonstrates the dramatic improvement in performance achieved through chain-of-thought prompting. Without CoT, even large language models struggle with multi-step reasoning problems. With CoT prompting, performance improved by 20-40 percentage points across various model sizes, highlighting how the right prompting strategy can unlock capabilities already present in the model.

6.2 Components: Instruction, Context, Input Data, Output Indicator

Effective prompts typically include:

Prompt Components

1. **Instruction:** Clear directions about the task
2. **Context:** Background information or constraints
3. **Input Data:** The specific content to process
4. **Output Indicator:** Format or style specifications

Example of a Structured Prompt:

Listing 5: Structured Prompt Components

```
# INSTRUCTION
Summarize the following medical research abstract in simple terms that a patient

# CONTEXT
This is for a patient education website. The audience has no medical background.

# INPUT DATA
[Research abstract text here]

# OUTPUT INDICATOR
Your summary should be 3-5 short paragraphs. Include a one-sentence "Key Takeawa
```

Case Study: Legal Document Analysis

Law firms use structured prompts to extract specific information from contracts. By providing clear instructions (e.g., "Identify all payment terms and obligations"), relevant context (e.g., "This is for a procurement contract review"), specific input data (the contract text), and output indicators (e.g., "Format as a table with clause references"), they achieve consistent, structured outputs that can be directly incorporated into legal workflows, demonstrating how well-crafted prompts can turn LLMs into specialized information extraction tools.

7 Future Developments

7.1 Agentic AI and Autonomous Agents

Agentic AI systems combine LLMs with:

- **Planning:** Breaking down complex goals into subtasks
- **Memory:** Maintaining information across interactions
- **Tool Use:** Leveraging external capabilities (APIs, databases, etc.)
- **Self-Improvement:** Learning from successes and failures

$$\text{Agent Architecture} = \begin{cases} \text{Perception Module} \\ \text{Memory System} \\ \text{Planning Engine} \\ \text{Action Execution} \\ \text{Learning Mechanism} \end{cases} \tag{31}$$

Case Study: BabyAGI

BabyAGI demonstrates simple but powerful agentic capabilities. Given a high-level task like "Research investment opportunities in renewable energy," it autonomously creates subtasks, executes them in a reasonable order, utilizes tools like web search and document analysis, and compiles findings into a coherent output. While limited compared to human researchers, its ability to work autonomously toward complex goals illustrates the direction of agent-based AI systems.

7.2 Debates on AI Capabilities and Potential Risks

7.2.1 Alignment and Safety

As AI systems become more capable, ensuring they act in accordance with human values becomes increasingly important:

$$\text{Alignment Gap} = \text{AI Capability} - \text{Alignment Level} \tag{32}$$

7.2.2 Scaling Laws and Emergent Abilities

Research suggests that capabilities may emerge non-linearly as models scale:

$$\text{Performance} \approx C \cdot (\text{Compute})^{\alpha} \cdot (\text{Data})^{\beta} \cdot (\text{Parameters})^{\gamma} \tag{33}$$

Case Study: Frontier Model Research

Research by organizations like Anthropic on frontier models has revealed surprising emergent capabilities. As models scaled beyond certain thresholds, they suddenly demonstrated abilities not observed in smaller versions, such as multi-step reasoning, code generation, and creative problem-solving. These discontinuous improvements suggest that further scaling may unlock additional capabilities that are difficult to predict in advance, highlighting both the potential and uncertainty in continued AI advancement.

8 Evaluation of LLMs

8.1 Code-based, Human Evaluation, LLM as Judge

8.1.1 Code-based Evaluation

Automated metrics provide objective but limited assessment:

- **BLEU, ROUGE:** Lexical overlap with reference texts
- **Perplexity:** Probability assigned to correct tokens
- **Task-specific Metrics:** Accuracy, F1 score, etc.

8.1.2 Human Evaluation

Human judgments capture nuanced quality aspects:

$$\text{Human Evaluation} = \begin{cases} \text{Helpfulness} \\ \text{Accuracy} \\ \text{Safety} \\ \text{Quality} \\ \text{Bias} \end{cases} \tag{34}$$

8.1.3 LLM as Judge

Using stronger models to evaluate outputs:

Listing 6: LLM-as-Judge Prompt Template

```
Rate the quality of the following response to the given query:

Query: [User query]
Response: [Model response]

Score from 1-10 on:
- Relevance to query
- Factual accuracy
- Completeness
- Clarity
- Helpfulness

Provide justification for each score.
```

Case Study: MMLU Benchmark

The Massive Multitask Language Understanding (MMLU) benchmark evaluates models across 57 subjects ranging from elementary mathematics to professional medicine. This comprehensive evaluation reveals both strengths and

weaknesses in model capabilities across different domains of knowledge. Recent models achieve human expert-level performance in some categories while still struggling in others, providing a nuanced picture of progress and remaining challenges in language model development.

8.2 Concepts like Distillation and Mixture of Experts

8.2.1 Knowledge Distillation

Transferring knowledge from larger to smaller models:

$$L_{\text{distill}} = \alpha L_{\text{task}} + (1 - \alpha) L_{\text{KD}} \tag{35}$$

Where L_{KD} measures the divergence between student and teacher model outputs.

8.2.2 Mixture of Experts (MoE)

Combining specialized sub-networks:

$$y = \sum_{i=1}^{n} g(x, i) \cdot f_i(x) \tag{36}$$

Where $g(x, i)$ is a gating function determining how much expert f_i contributes to the output.

Case Study: Google's Switch Transformer

Google's Switch Transformer demonstrated the efficiency gains possible with MoE architectures. By using a sparse mixture of experts approach where only a subset of experts process each input token, the model achieved performance comparable to dense models with significantly less computation during inference. This approach enables larger effective model sizes while maintaining reasonable training and deployment costs, potentially offering a more efficient scaling path than simply increasing dense model parameters.

Benefits of MoE:

- Computational efficiency through sparse activation
- Specialization of different components for different subtasks
- Capacity scaling without proportional computation increase

$$\text{Parameters in MoE} \gg \text{Parameters used per forward pass} \tag{37}$$

$$\text{Effective Capacity} \approx \text{Experts} \times \text{Parameters per Expert} \tag{38}$$

$$\text{Computation} \approx \text{Active Experts} \times \text{Parameters per Expert} \tag{39}$$