Cognition and Computation

Comprehensive Course Notes

Based on lectures by Prof. Marco Zorzi and Dr. Alberto Testolin
University of Padova

Academic Year 2024-2025

Abstract

These notes provide a comprehensive overview of the "Cognition and Computation" course, covering computational approaches to modeling cognitive functions, from artificial neural networks to probabilistic models. The material is relevant for understanding both how the mind works and for developing and evaluating modern artificial intelligence systems. The theoretical discussion of different approaches is accompanied by concrete application examples in vision, language processing, and reasoning.

Contents

1 Introduction to Computational Cognitive Modeling **6**
- 1.1 What is Cognitive Modeling and Its Relation to AI 6
  - 1.1.1 Leading Computational Frameworks 6
  - 1.1.2 Definition of Intelligence and Turing Test 7
- 1.2 Methodological Issues in Cognitive Modeling 7
  - 1.2.1 From Theory to Computational Model 7
  - 1.2.2 Evaluation of Computational Models 8
  - 1.2.3 Adjudicating Between Competing Models 8

2 Philosophical Perspectives on the Mind/Brain Problem **9**
- 2.1 Dualism vs. Monism 9
- 2.2 Thought Experiments on the Mind/Brain Problem 9
- 2.3 The Source of Human Knowledge 10
- 2.4 Strong AI: Can Machines Really Think? 10
- 2.5 Computational Complexity and Rational Agents 11

3 Probabilistic Graphical Models: Representation **12**
- 3.1 The Issue of Knowledge Representation 12
- 3.2 Benefits of Structured Representations 13
- 3.3 Bayesian Networks 13
  - 3.3.1 Structure and Factorization 13
  - 3.3.2 Markov Blanket 13
  - 3.3.3 Example: Student Grade Model 14
- 3.4 Markov Networks 14
  - 3.4.1 Structure and Factorization 14
  - 3.4.2 Markov Blanket in Markov Networks 14
  - 3.4.3 Comparison with Bayesian Networks 15

4 Probabilistic Graphical Models: Inference **16**
- 4.1 Reasoning Patterns in Bayesian Networks 16
- 4.2 Inference Algorithms 16
  - 4.2.1 Exact Inference 17
  - 4.2.2 Approximate Inference 17

4.3 Gibbs Sampling 17

5 Probabilistic Graphical Models: Learning **19**
5.1 Learning in PGMs 19
5.2 Maximum Likelihood Estimation 19
5.3 Structure Learning 20
5.4 Learning with Hidden Variables 20
5.5 Causal Inference 21

6 Probabilistic Programming **22**
6.1 From Graphical Models to Probabilistic Programming 22
6.2 Probabilistic Programming Languages (PPLs) 22
6.3 Bayesian Program Learning 23
6.4 Deep Probabilistic Programming 23

7 Neural Computation **24**
7.1 Basic Principles of Neural Computation 24
7.2 The Neuron Model 24
7.3 Network Architectures 25
7.4 Learning in Neural Networks 25
7.5 Historical Development of Neural Networks 26

8 Supervised Learning in Neural Networks **27**
8.1 The Perceptron 27
8.1.1 Perceptron Learning Rule 27
8.1.2 Limitations of the Perceptron 28
8.2 The Delta Rule 28
8.3 Multi-layer Networks and Backpropagation 28
8.4 Learning Parameters and Regularization 29

9 Supervised Deep Learning and Convolutional Neural Networks **30**
9.1 Deep Feedforward Networks 30
9.2 Convolutional Neural Networks (CNNs) 31
9.2.1 Key Components 31
9.2.2 Historical Development of CNNs 31
9.3 CNN Relation to Visual Processing in the Brain 32
9.4 Limitations of CNNs 32

10 Unsupervised Deep Learning **33**
10.1 Principles of Unsupervised Learning 33
10.2 Dimensionality Reduction 33
10.3 Representation Learning with Deep Networks 34
10.4 Restricted Boltzmann Machines (RBMs) 34
10.5 Deep Belief Networks (DBNs) 34

10.6 Generative Adversarial Networks (GANs) . . . 35
10.7 Variational Autoencoders (VAEs) . . . 35
10.8 Diffusion Models . . . 35

11 Reinforcement Learning **36**
11.1 Basics of Reinforcement Learning . . . 36
11.2 Markov Decision Processes . . . 36
11.3 Value Functions and the Bellman Equation . . . 37
11.4 Temporal Difference Learning . . . 37
11.5 Q-Learning . . . 38
11.6 Deep Reinforcement Learning . . . 38
11.7 Reinforcement Learning in the Brain . . . 38

12 Sequence Learning and Self-Supervised Learning **40**
12.1 Learning Temporal Dependencies . . . 40
12.1.1 Approaches to Sequence Learning . . . 40
12.2 Simple Recurrent Networks (SRNs) . . . 40
12.3 Self-Supervised Learning . . . 41
12.4 Long Short-Term Memory (LSTM) Networks . . . 41
12.5 Attention Mechanisms and Transformers . . . 41
12.6 Other Self-Supervised Learning Approaches . . . 42

13 Natural Language Processing **43**
13.1 Evolution of NLP Methods . . . 43
13.2 Statistical Language Models . . . 43
13.3 Word Embeddings . . . 44
13.4 Transformer-Based Models . . . 44
13.5 Large Language Models (LLMs) . . . 45
13.6 Text Generation and Decoding Methods . . . 45

14 Learning and Memory Systems **46**
14.1 Types of Memory in Neural Networks . . . 46
14.2 Catastrophic Forgetting and Complementary Learning Systems 46
14.3 Continual Learning in Neural Networks . . . 47
14.4 Spontaneous Brain Activity and Generative Models . . . 47

15 Towards Artificial General Intelligence (AGI) **49**
15.1 The Concept of AGI . . . 49
15.2 Defining and Assessing AGI . . . 49
15.3 The Grounding Problem and Embodied Cognition . . . 50
15.4 Scaling Laws and Emergent Abilities . . . 50
15.5 Current Capabilities and Limitations . . . 51

16 Perspectives and Challenges for Deep Learning AI **52**
16.1 Robustness Challenges 52
16.2 Explainable AI (XAI) 52
16.3 AI Fairness and Ethics 53
16.4 Regulatory Approaches 53
16.5 Future Directions 54
16.6 Conclusion 54

Chapter 1

Introduction to Computational Cognitive Modeling

1.1 What is Cognitive Modeling and Its Relation to AI

Cognitive modeling refers to the development of computational models that simulate human cognitive processes. While it shares methodologies with artificial intelligence (AI), the focus differs significantly. AI primarily aims to build intelligent systems regardless of whether they mirror human cognition, whereas cognitive modeling explicitly attempts to reproduce human cognitive mechanisms.

1.1.1 Leading Computational Frameworks

Throughout the history of cognitive science, two major approaches have competed to explain human cognition:

1. **Symbolic Approach (Cognitivism)**: Championed by researchers like Newell, Simon, McCarthy, Minsky, and Chomsky, this framework views the mind as a computational system that manipulates symbols according to formal rules. The symbolic approach focuses on higher-level cognition like reasoning, problem-solving, and language.

2. **Emergentist Approach (Connectionism)**: Developed by researchers like McCulloch, Wiener, McClelland, Hinton, and Grossberg, this framework models cognition as emerging from the interaction of simple neural-like units. The emergentist approach excels at modeling perception, pattern recognition, and learning from experience.

The symbolic approach dominated early AI (1960s-1970s), with systems that could solve logic problems and engage in simple dialogues. However, it struggled with tasks that humans find intuitive, like vision and natural language understanding, leading to the first "AI winter."

The connectionist approach gained prominence in the 1980s with the development of backpropagation and multi-layer networks, showing promise in learning from experience. However, limitations in computing power and data availability led to a second decline in the 1990s.

Modern approaches have evolved into:

- **Structured Probabilistic Models**: The contemporary symbolic approach incorporating uncertainty
- **Deep Learning**: The modern emergentist approach with multiple layers of representation

1.1.2 Definition of Intelligence and Turing Test

The definition of intelligence remains a subject of debate. Wikipedia defines it as "the capacity for logic, understanding, self-awareness, learning, emotional knowledge, reasoning, planning, creativity, critical thinking, and problem-solving." More generally, it involves perceiving information, retaining it as knowledge, and applying it to adaptive behaviors.

Alan Turing proposed the "imitation game" (now known as the Turing Test) to determine if a machine can be considered intelligent. In this test, a human evaluator converses with both a machine and another human. If the evaluator cannot reliably distinguish between the machine and the human, the machine is considered to have passed the test and demonstrated intelligence.

1.2 Methodological Issues in Cognitive Modeling

1.2.1 From Theory to Computational Model

Developing a computational model involves translating theoretical assumptions into precise algorithms and representations. This process often reveals areas where the initial theory is incomplete or underspecified. Many modeling assumptions may remain implicit—making them explicit and assessing their importance is crucial for understanding a model's successes and limitations.

1.2.2 Evaluation of Computational Models

Descriptive Adequacy

Models can be evaluated based on how well they capture human behavioral patterns:

- **Qualitative assessment**: Does the model exhibit the same effects as humans? For example, in numerosity comparison tasks, do both humans and the model show similar performance patterns based on numerical ratio?
- **Quantitative assessment**: How closely does the model's performance match human data? This can be assessed through statistical measures like correlation between model and human performance.

Other Evaluation Criteria

Beyond descriptive adequacy, models can be judged on:

- **Generality**: Can the model apply to different tasks, stimuli, and conditions?
- **Simplicity/Parsimony**: Is the model more parsimonious than alternatives with similar explanatory power?
- **Transparency**: Can we understand how the model works, or is it a "black box"?

1.2.3 Adjudicating Between Competing Models

When different models make similar predictions, researchers must use additional criteria to choose between them:

1. Compare descriptive adequacy: Can one model be falsified by showing it fails to reproduce some behavioral effect?
2. Apply the simplicity criterion: When models perform similarly, prefer the simpler one.
3. Consider explanatory value: A model has explanatory value when its assumptions are explicit, theoretically justified, and empirically tested.

Chapter 2

Philosophical Perspectives on the Mind/Brain Problem

2.1 Dualism vs. Monism

The relationship between mind and brain has been debated for centuries, with two main philosophical positions:

- **Dualism**: Championed by René Descartes (1596-1650), this view posits a clear distinction between the mental realm (res cogitans) including phenomena like memory, reasoning, and consciousness, and the physical realm (res extensa) comprising processes like neural transmission and physiological mechanisms. A key challenge for dualism is explaining how these realms interact.
- **Monism**: This view argues that mental and physical processes are essentially the same thing. Physicalism claims there is only matter (no mind), while idealism claims there is only mind (no matter). Monism must explain how thinking emerges from physical processes or vice versa.

2.2 Thought Experiments on the Mind/Brain Problem

Philosophers have devised several thought experiments to explore the relationship between mind and brain:

1. **Brain in a Vat**: If your brain were removed at birth and placed in a vat with simulated inputs, would your mental states be identical to someone experiencing the actual world? This explores whether mental states are determined solely by brain states.

2. **Avicenna's "Floating Man"**: Avicenna proposed imagining a person created instantly and floating in air with no sensory input. Would this person be self-aware despite having no sensory experience? This questions whether self-consciousness requires sensory knowledge.

3. **Brain Replacement**: If neurons in your brain were gradually replaced with functionally identical artificial components, at what point would you cease to be conscious? This probes the physical basis of consciousness.

2.3 The Source of Human Knowledge

Another fundamental philosophical debate concerns how humans acquire knowledge:

- **Innatism**: Associated with Plato, this view holds that we are born with innate ideas, concepts, and principles. Sense-perception merely activates or completes these innate structures. Knowledge is primarily discovered through deduction.

- **Empiricism**: Associated with Aristotle, this view holds that we begin as a blank slate (tabula rasa) and acquire knowledge through sensory experience. Learning occurs by discovering statistical regularities in the world, and knowledge is primarily acquired through induction.

- **Modern Synthesis**: Contemporary cognitive science suggests a trade-off—knowledge develops through learning and experience, but innate biases and constraints guide learning toward useful patterns. This relates to the "no free lunch theorem" in machine learning, which states that no learning algorithm can succeed without some inductive bias.

2.4 Strong AI: Can Machines Really Think?

The question of whether machines can genuinely think rather than merely simulate thinking remains controversial.

John Searle's "Chinese Room" thought experiment challenges the idea that passing the Turing test demonstrates understanding. In this scenario, a person who doesn't understand Chinese follows rules for manipulating Chinese symbols, producing appropriate responses without understanding the meaning. Searle argues this shows that syntactic rule-following (computation) cannot produce semantic understanding.

Critics counter that:

- The system as a whole might understand even if the individual following the rules doesn't

- Embodied systems with causal connections to the world might achieve "symbol grounding"

2.5 Computational Complexity and Rational Agents

Philosophical debates about AI must consider practical computational constraints:

- Many problems involve combinatorial explosion, making naive approaches computationally intractable
- Real machines (and brains) operate under constraints of memory (space complexity) and time (temporal complexity)
- Perfect rationality (always doing the optimal thing) is often unachievable in complex environments
- Rational process models should consider the cognitive resources available to the agent

Chapter 3

Probabilistic Graphical Models: Representation

3.1 The Issue of Knowledge Representation

How can we represent different hypotheses or concepts in cognitive models? Two major approaches exist:

1. **Structured (Localist) Representations**:
    - Separate variables represent each hypothesis
    - Explicit graph structure defines how hypotheses interact
    - Knowledge is represented declaratively, separating it from reasoning
    - Facilitates interpretability
    - Challenge: determining which variables to include and how to structure them
2. **Distributed (Sub-symbolic) Representations**:
    - Each hypothesis is represented by a pattern of activation across many units
    - The system learns interaction dynamics through experience
    - Knowledge is implicit in connection weights
    - Knowledge and reasoning are not separated
    - Challenge: interpreting what the system has learned (potential "black box")

3.2 Benefits of Structured Representations

Structured explicit representations offer several advantages:

- Support "few-shot" learning (adaptation with limited examples)
- Naturally allow for compositionality (building complex concepts from simpler ones)
- Can incorporate innate core knowledge as inductive biases:
  - Intuitive physics (understanding of physical objects)
  - Intuitive psychology (understanding of agents' goals and beliefs)
  - Generative grammars (rule systems for language)

3.3 Bayesian Networks

3.3.1 Structure and Factorization

Bayesian networks are directed acyclic graphs where:

- Nodes represent random variables
- Edges represent "parent of" relationships (causal influence)
- Each variable has an associated conditional probability distribution (CPD)
- The joint distribution factorizes as the product of local CPDs:

$$P(A, B, C, ..., H) = P(A)P(B|A)P(C|A)P(D|B, C)P(E|D, G)...P(H|E) \tag{3.1}$$

Bayesian networks efficiently represent joint probability distributions. For n binary variables, the full joint distribution would require $2^n - 1$ parameters, but factorization reduces this number dramatically.

3.3.2 Markov Blanket

The Markov blanket of a node is the set of nodes that, when observed, make the node independent from all others in the network. It consists of:

- The node's parents
- The node's children
- The parents of the node's children

Knowing the values of all variables in a node's Markov blanket makes that node conditionally independent of all other variables in the network.

3.3.3 Example: Student Grade Model

Consider a simple Bayesian network modeling student performance:

- Intelligence (I): high or low
- Difficulty (D): easy or hard course
- SAT Score (S): high or low, depending on Intelligence
- Grade (G): depends on both Intelligence and course Difficulty
- Letter of Recommendation (L): depends only on Grade

This network encodes conditional independencies:

- SAT and Grade are conditionally independent given Intelligence
- Letter is conditionally independent of Intelligence and Difficulty given Grade

The joint distribution factorizes as:

$$P(I, D, S, G, L) = P(I)P(D)P(S|I)P(G|I, D)P(L|G) \tag{3.2}$$

3.4 Markov Networks

3.4.1 Structure and Factorization

Markov networks are undirected graphical models where:

- Nodes represent random variables
- Edges represent "affinity" or correlation between variables
- Each edge has an associated factor indicating correlation strength
- The joint distribution factorizes as a product of local factors:

$$P(A, B, C, ..., H) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi \tag{3.3}$$

where Z is a normalization constant and Φ is the set of all factors

3.4.2 Markov Blanket in Markov Networks

The Markov blanket of a node in a Markov network is simply its immediate neighbors, making the structure more intuitive in some ways than Bayesian networks.

3.4.3 Comparison with Bayesian Networks

- Bayesian networks specify clear semantic relationships and can encode causal relationships
- Markov networks can flexibly encode "affinity" between variables without specifying explicit probability distributions
- Bayesian networks are often better for models built with expert knowledge
- Markov networks are commonly used in learning models, though their semantics may be less transparent

Chapter 4

Probabilistic Graphical Models: Inference

4.1 Reasoning Patterns in Bayesian Networks

Bayesian networks support three fundamental reasoning patterns:

1. **Causal Reasoning**: Reasoning from causes to effects (top-down in the graph)
    - Example: "If the student has low intelligence, how likely is a poor grade?"
2. **Evidential Reasoning**: Reasoning from effects to causes (bottom-up in the graph)
    - Example: "If the student received a poor grade, how likely is low intelligence?"
3. **Intercausal Reasoning**: Reasoning between causes of a common effect
    - Example: "If the student got a poor grade but high SAT score, how does this affect our belief about course difficulty?"
    - This involves "explaining away" - when one cause becomes more likely, alternative causes may become less likely

4.2 Inference Algorithms

Inference in probabilistic graphical models involves computing conditional probabilities:

$$P(y|e) = \frac{P(y,e)}{P(e)} = \frac{P(y,e)}{\sum_y P(y,e)} \tag{4.1}$$

where y represents query variables and e represents evidence (observed variables).

4.2.1 Exact Inference

- **Inference by Enumeration**: Compute the joint probability, summing out variables that are neither query nor evidence
- **Variable Elimination**: More efficient algorithm that exploits factorization to compute sums incrementally
- **Belief Propagation**: Efficient for tree-structured networks, involves passing messages between nodes

However, exact inference becomes computationally intractable for complex networks.

4.2.2 Approximate Inference

- **Sampling Methods**: Generate samples from the distribution to estimate probabilities
  - **Rejection Sampling**: Generate samples from the full distribution, keeping only those consistent with evidence
  - **Importance Sampling**: Generate samples focusing on regions of high probability given evidence
  - **Markov Chain Monte Carlo (MCMC)**: Generate samples by constructing a Markov chain that converges to the target distribution
- **Variational Methods**: Approximate the true distribution with a simpler one that can be computed analytically

4.3 Gibbs Sampling

Gibbs sampling is a widely used MCMC method:

1. Start with random values for non-evidence variables
2. Repeatedly sample one variable at a time, conditioned on the current values of all other variables
3. After a "burn-in" period, the samples approximate the true posterior distribution

Gibbs sampling is efficient because:

- Variables need only be conditioned on their Markov blanket, not the entire network
- Conditionally independent variables can be sampled simultaneously (block Gibbs sampling)

Chapter 5

Probabilistic Graphical Models: Learning

5.1 Learning in PGMs

Learning in probabilistic graphical models can involve:

- Parameter learning: estimating values in CPD tables given a fixed structure
- Structure learning: determining the graph structure from data
- Learning with latent variables: inferring both structure and parameters when some variables are unobserved

5.2 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) finds the parameter values that maximize the probability of observing the training data:

$$\theta_{MLE} = \arg\max_{\theta} P(D|\theta) \tag{5.1}$$

where D is the training data and θ represents model parameters.

For Bayesian networks, the likelihood function decomposes according to the network structure, allowing parameters for each CPD to be estimated independently:

$$L(\theta; D) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \tag{5.2}$$

where:

- θ_{ijk} is the probability of variable i taking value k given parent configuration j

- N_{ijk} is the count of instances where variable i takes value k with parent configuration j
- q_i is the number of possible parent configurations for variable i
- r_i is the number of possible values for variable i

For multinomial distributions (e.g., discrete Bayesian networks), the MLE is simply the empirical frequency:

$$\theta_{ijk} = \frac{N_{ijk}}{\sum_k N_{ijk}} \tag{5.3}$$

5.3 Structure Learning

Learning network structure is more challenging:

- The space of possible structures grows super-exponentially with the number of variables
- Basic approach:
  - Start with minimal or random structure
  - Iteratively improve by adding, deleting, or reversing edges
  - Compute a score for each structure (e.g., likelihood penalized for complexity)
  - Select the highest-scoring structure
- Search strategies include greedy search, simulated annealing, and genetic algorithms

5.4 Learning with Hidden Variables

When some variables are unobserved, learning becomes more complex:

- Expectation-Maximization (EM) algorithm is commonly used
- E-step: Estimate distribution of hidden variables given current parameters
- M-step: Update parameters to maximize expected likelihood given hidden variable distributions
- Repeat until convergence

5.5 Causal Inference

Distinguishing correlation from causation is a fundamental challenge. Approaches include:

- **Randomized Experiments**: Intervene by randomly assigning subjects to treatment groups
- **Counterfactual Reasoning**: Use the do-calculus to reason about hypothetical interventions
  - $P(Y|do(X = x))$ represents the distribution of Y if we intervene to set $X = x$, different from the conditional probability $P(Y|X = x)$

Chapter 6

Probabilistic Programming

6.1 From Graphical Models to Probabilistic Programming

Probabilistic programming combines probabilistic modeling with programming concepts:

- Instead of representing models as graphs, they are expressed as computer programs
- Programs can use constructs like loops, recursion, and conditionals, which are difficult to represent in finite graphs
- The output of a probabilistic program is a probability distribution

6.2 Probabilistic Programming Languages (PPLs)

PPLs extend standard programming languages with probabilistic constructs:

- Variables can be random with associated distributions
- Learning involves constructing programs that best explain observations under Bayesian criteria
- Inference engines automatically determine parameter distributions given observations
- Hierarchical priors allow transfer of knowledge between related concepts

6.3 Bayesian Program Learning

One example of probabilistic programming is Bayesian Program Learning (BPL) for one-shot character recognition:

- Characters are represented as structured, generative programs
- Each character type is defined by strokes and spatial relations between strokes
- The joint distribution includes a prior over program structure and parameters
- Learning involves finding program parameters that best fit observed examples
- Inference uses MCMC to search the space of programs that could generate an observed image

BPL outperforms deep learning models on one-shot classification tasks where humans see only a single example of a new character and must identify it among similar characters.

6.4 Deep Probabilistic Programming

Recent developments combine deep learning with probabilistic programming:

- Exploit deep learning to speed up inference
- Incorporate distributed representations of knowledge
- Examples include Edward, Pyro, and Infer.NET

These approaches aim to combine the flexibility and interpretability of probabilistic programming with the pattern recognition capabilities of deep learning.

Chapter 7

Neural Computation

7.1 Basic Principles of Neural Computation

Neural computation is information processing carried out by networks of neurons, artificial or biological. Key principles include:

- **Neurons**: Serve as detectors that signal with their activity
- **Networks**: Link, coordinate, and select patterns of activity
- **Learning**: Organizes networks to perform tasks and develop internal models

Compared to traditional computing:

- Memory and processing are integrated rather than separated
- Operations involve detection, weighting evidence, and evaluation rather than logic and arithmetic
- Complex computations emerge from networks of specialized detectors rather than arbitrary sequences of operations

7.2 The Neuron Model

The formal neuron is a mathematical model capturing key aspects of biological neurons:

- Receives signals from other neurons through connections
- Each connection has a weight representing its strength (positive for excitatory, negative for inhibitory)

- The net input is the weighted sum of incoming signals:

$$net_i = \sum_{j=1}^{N} w_{ij} x_j - b \tag{7.1}$$

  where x_j is the signal from neuron j, w_{ij} is the connection weight, and b is a bias term

- The activation is computed by applying a non-linear function to the net input:

$$o_i = f(net_i) \tag{7.2}$$

- Common activation functions include sigmoid, hyperbolic tangent, and rectified linear unit (ReLU)

7.3 Network Architectures

Neural networks are organized into different architectures:

1. **Pattern Associator**: Input and output layers with unidirectional connections
2. **Multi-layer Feedforward Network**: Include one or more hidden layers between input and output, with unidirectional connections flowing forward
3. **Recurrent Network**: Allows activation to flow backward (feedback connections)
4. **Fully Recurrent Network**: Includes both feedback and lateral connections within layers

7.4 Learning in Neural Networks

Learning in neural networks involves adjusting connection weights based on experience:

- **Hebbian Learning**: Strengthens connections between simultaneously active neurons

$$\Delta w_{ij} = \eta y_i x_j \tag{7.3}$$

  where η is the learning rate, y_i is the activation of the receiving neuron, and x_j is the activation of the sending neuron

- **Supervised Learning**: Uses explicit teaching signals or desired outputs

- **Unsupervised Learning**: Extracts statistical regularities without external feedback
- **Reinforcement Learning**: Guided by reward signals indicating success or failure

7.5 Historical Development of Neural Networks

Neural networks have undergone several phases of development:

- **1943-1960**: Early concepts including McCulloch-Pitts neuron, Hebbian learning, and Rosenblatt's perceptron
- **1969-1981**: Decline following Minsky and Papert's critique of perceptrons
- **1982-1990**: Renaissance with Hopfield networks, backpropagation learning, and the PDP (Parallel Distributed Processing) framework
- **1998-2005**: Further developments in reinforcement learning and generative models
- **2006-present**: Deep learning revolution and widespread application

Chapter 8

Supervised Learning in Neural Networks

8.1 The Perceptron

The perceptron, developed by Frank Rosenblatt in 1958, was the first neural network with modifiable weights:

- Single output neuron computing:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} w_i x_i - \theta \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{8.1}$$

  where θ is a threshold

- The threshold can be replaced by a bias term w_0 with a fixed input $x_0 = 1$:

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{N} w_i x_i \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{8.2}$$

8.1.1 Perceptron Learning Rule

- Training examples consist of input vectors x and desired outputs t (targets)
- If output matches target ($y = t$), no weight change occurs
- If output differs from target ($y \neq t$), weights are updated:

$$\Delta w_i = \eta t x_i \tag{8.3}$$

  where η is the learning rate

- The Perceptron Convergence Theorem guarantees that if a problem is linearly separable, the perceptron will find a solution in a finite number of steps

8.1.2 Limitations of the Perceptron

- Can only solve linearly separable problems
- Cannot learn XOR or other nonlinear functions

8.2 The Delta Rule

The delta rule extends the perceptron learning rule to neurons with continuous activation functions:

- Uses a cost function based on squared error:

$$E = \frac{1}{2}\sum_i (t_i - y_i)^2 \tag{8.4}$$

- Weight updates follow the negative gradient of the cost function:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{8.5}$$

- For linear activation functions, this simplifies to:

$$\Delta w_{ij} = \eta(t_i - y_i)x_j \tag{8.6}$$

- For sigmoid activation functions:

$$\Delta w_{ij} = \eta(t_i - y_i)y_i(1 - y_i)x_j \tag{8.7}$$

8.3 Multi-layer Networks and Backpropagation

Multi-layer networks can learn nonlinear functions but require a method to train hidden layers:

- Error backpropagation extends the delta rule to networks with hidden layers
- Forward pass: Compute activations through the network
- Backward pass: Propagate error signals from output to earlier layers
- For output neurons, the error signal (delta) is:

$$\delta_i = (t_i - y_i)f'(net_i) \tag{8.8}$$

- For hidden neurons, the error signal is:

$$\delta_j = f'(net_j)\sum_i w_{ij}\delta_i \tag{8.9}$$

- Weight updates are then:

$$\Delta w_{ji} = \eta \delta_j h_i \tag{8.10}$$

  where h_i is the activation of the sending neuron

8.4 Learning Parameters and Regularization

Several parameters and techniques affect learning:

- **Learning Rate**: Controls step size during weight updates
  - Small values: slow learning but stable
  - Large values: faster learning but risk of overshooting
  - Adaptive learning rates decrease over time
- **Momentum**: Adds a fraction of the previous weight update to the current one

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) + \eta \Delta w_{ij}(t) \tag{8.11}$$

  where α is the momentum term (typically 0.7-0.9)
- **Regularization**: Techniques to prevent overfitting
  - Weight decay (L2 regularization): Penalizes large weights
  - L1 regularization: Promotes sparsity
  - Dropout: Randomly deactivates neurons during training
  - Early stopping: Halts training when validation error begins to increase

Chapter 9

Supervised Deep Learning and Convolutional Neural Networks

9.1 Deep Feedforward Networks

Deep learning involves neural networks with many hidden layers, allowing hierarchical processing where information becomes increasingly abstract at deeper levels.

Key advantages of deep learning:

- End-to-end learning without manual feature engineering
- Automatic discovery of useful representations
- Ability to model complex, non-linear relationships

Challenges and solutions:

- **Vanishing Gradient**: As error is backpropagated through many layers, the gradient can become extremely small
  - Solution: New activation functions like ReLU ($f(x) = \max(0, x)$) which don't saturate for positive inputs
- **Overfitting**: Increased model complexity raises the risk of fitting to noise
  - Solutions: Stronger regularization, sparse connectivity, large training sets

9.2 Convolutional Neural Networks (CNNs)

CNNs are biologically inspired deep networks designed for processing grid-like data (e.g., images):

9.2.1 Key Components

1. **Convolutional Layers**:
    - Neurons have local receptive fields (sparse connectivity)
    - Same filter is applied across the entire input (weight sharing)
    - Multiple filters extract different features, creating feature maps
    - Reduces parameters compared to fully connected networks
2. **Pooling Layers**:
    - Reduce spatial dimensions by summarizing regions
    - Common operations: max pooling, average pooling
    - Provide translation invariance
    - Control overfitting by reducing parameters
3. **Fully Connected Layers**:
    - Typically in the deepest parts of the network
    - Integrate information from convolutional layers
    - Perform classification based on extracted features

9.2.2 Historical Development of CNNs

- **Neocognitron** (Fukushima, 1980): Early hierarchical model inspired by visual cortex
- **LeNet** (LeCun et al., 1998): First CNN with learnable weights, used for digit recognition
- **AlexNet** (Krizhevsky et al., 2012): Breakthrough in ImageNet competition, reducing error rate by 11%
- **VGG** (Simonyan Zisserman, 2014): Deeper network with uniform architecture
- **GoogLeNet/Inception** (Szegedy et al., 2015): Used inception modules with parallel convolutions
- **ResNet** (He et al., 2015): Introduced residual connections to train very deep networks

9.3 CNN Relation to Visual Processing in the Brain

CNNs share organizational principles with the visual system:

- Hierarchical processing from simple to complex features
- Ventral pathway ("What"): Object recognition
- Dorsal pathway ("Where"): Spatial information and motion
- Representational Similarity Analysis (RSA) shows correspondence between CNN layers and brain areas

9.4 Limitations of CNNs

Despite their success, CNNs have important limitations:

- **Vulnerability to Noise**: Performance degrades significantly with image noise or distortions
- **Sensitivity to Viewpoint**: Fail to recognize objects from unusual angles
- **Susceptibility to Adversarial Examples**: Small, carefully crafted perturbations can cause misclassification
- **Lack of Explainability**: Difficult to interpret what features drive decisions

These limitations highlight differences between CNNs and human vision, which is more robust to perturbations and generalizes better across contexts.

Chapter 10

Unsupervised Deep Learning

10.1 Principles of Unsupervised Learning

Unsupervised learning involves extracting structure from data without explicit labels:

- **Advantages**:
  - Doesn't require labeled data
  - Focuses on representation learning (feature extraction)
  - Biologically plausible, especially for developmental learning
  - Can use biologically plausible learning rules
- **Challenges**:
  - Determining which features will be useful
  - Defining what constitutes a "good" representation

10.2 Dimensionality Reduction

Dimensionality reduction techniques project high-dimensional data into a lower-dimensional space:

- **Principal Component Analysis (PCA)**:
  - Finds directions of maximum variance in the data
  - Projects data onto these principal components
  - Retains components explaining sufficient variance
  - Linear transformation that minimizes reconstruction error

10.3 Representation Learning with Deep Networks

Deep networks excel at learning hierarchical representations:

- Lower layers capture basic features, higher layers synthesize complex patterns
- Learning is typically unsupervised, discovering patterns without labels
- Pre-trained representations can be transferred to new tasks (transfer learning)
- Training requires careful regularization to avoid learning trivial features

10.4 Restricted Boltzmann Machines (RBMs)

RBMs are stochastic neural networks that learn generative models of data:

- Bipartite graph with visible units (data) and hidden units (features)
- No connections within layers (restricted Boltzmann machine)
- Energy function defines probability of configurations:

$$P(v, h) = \frac{1}{Z} e^{-E(v,h)} \tag{10.1}$$

  where $E(v, h) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_i a_i v_i - \sum_j b_j h_j$

- Contrastive Divergence (CD) learning approximates maximum likelihood:
  - Positive phase: Compute hidden activations given data
  - Negative phase: Generate reconstructions by sampling
  - Adjust weights to reduce discrepancy between data and model distributions

10.5 Deep Belief Networks (DBNs)

DBNs stack multiple RBMs to learn hierarchical representations:

- Trained layer-by-layer, treating each layer's outputs as inputs for the next
- Learn increasingly abstract features at deeper layers

- After unsupervised pre-training, the network can be fine-tuned with backpropagation for specific tasks
- Can be used for both generation (top-down) and recognition (bottom-up)

10.6 Generative Adversarial Networks (GANs)

GANs learn to generate realistic data through adversarial training:

- **Generator**: Network that creates fake data from random noise
- **Discriminator**: Network that distinguishes real from fake data
- Training involves a minimax game:
  - Generator tries to fool the discriminator
  - Discriminator tries to correctly classify real and fake samples
- At equilibrium, generator produces data indistinguishable from real samples

GANs have shown remarkable results in generating realistic images, though training can be unstable.

10.7 Variational Autoencoders (VAEs)

VAEs combine autoencoders with probabilistic modeling:

- Encoder maps inputs to a distribution in latent space
- Decoder reconstructs inputs from samples of the latent distribution
- Training maximizes a variational lower bound on data likelihood
- Enables both reconstruction and generation of new samples

10.8 Diffusion Models

Diffusion models learn to reverse a gradual noising process:

- Forward process: Gradually add noise to data
- Reverse process: Learn to denoise step by step
- Generation involves sampling noise and progressively denoising
- Currently produces state-of-the-art image generation results

Chapter 11

Reinforcement Learning

11.1 Basics of Reinforcement Learning

Reinforcement learning (RL) addresses the problem of learning how to behave optimally through trial and error:

- Agent learns to maximize cumulative rewards by interacting with an environment
- Differs from supervised learning because exact desired actions aren't known
- Must balance exploration (trying new actions) with exploitation (using known good actions)
- Historical roots in animal learning studies (classical and operant conditioning)

11.2 Markov Decision Processes

RL problems are typically formalized as Markov Decision Processes (MDPs):

- Set of states S
- Set of actions A
- Transition function $T(s_{t+1}|s_t, a_t)$
- Reward function $R(r_{t+1}|s_t, a_t)$
- Discount factor γ determining the importance of future rewards

The goal is to find a policy $\pi(a|s)$ that maximizes expected cumulative (discounted) reward:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{11.1}$$

11.3 Value Functions and the Bellman Equation

Value functions estimate the expected return from a state or state-action pair:

- State value function $V^\pi(s)$: Expected return starting from state s, following policy π

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right] \tag{11.2}$$

- Action value function $Q^\pi(s, a)$: Expected return after taking action a in state s, then following policy π

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right] \tag{11.3}$$

These value functions satisfy recursive Bellman equations:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma V^\pi(s')\right] \tag{11.4}$$

$$Q^\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a')\right] \tag{11.5}$$

11.4 Temporal Difference Learning

Temporal Difference (TD) learning uses experience to update value estimates:

- TD prediction learns value functions from experience
- Updates based on the difference between consecutive value estimates (TD error)

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right] \tag{11.6}$$

where α is the learning rate

11.5 Q-Learning

Q-learning is an off-policy TD algorithm that learns optimal action values directly:

- Updates Q-values based on the maximum Q-value in the next state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (11.7)$$

- Converges to optimal Q-values regardless of the policy followed during learning
- Usually paired with an exploration strategy like ϵ-greedy or softmax

11.6 Deep Reinforcement Learning

Deep RL combines RL algorithms with deep neural networks:

- **Deep Q-Network (DQN)**:
  - Uses neural networks to approximate Q-values
  - Includes stabilizing mechanisms:
    - Experience replay buffer to break correlations in sequential data
    - Target network updated periodically to reduce non-stationarity
  - Achieved human-level performance on Atari games
- **Policy Gradient Methods**:
  - Directly optimize policy parameters
  - Examples include REINFORCE, A3C, PPO
- **Actor-Critic Methods**:
  - Combine value function estimation with policy optimization
  - Examples include A2C, DDPG, SAC

11.7 Reinforcement Learning in the Brain

Evidence suggests RL mechanisms in the brain:

- Dopamine neurons in the midbrain signal reward prediction errors
- These prediction errors closely resemble TD errors

- These signals influence learning in neural circuits involved in motivation and decision-making
- Disruptions to this system are implicated in conditions like addiction, compulsive behavior, and Parkinson's disease

Chapter 12

Sequence Learning and Self-Supervised Learning

12.1 Learning Temporal Dependencies

Many cognitive tasks involve processing sequential information:

- Sensory stimulation occurs as a continuous stream
- Sequential information may contain temporal dependencies
- The brain extracts statistical structure over time (e.g., language regularities)

12.1.1 Approaches to Sequence Learning

- **Sliding Window**: Transform time into space by presenting multiple sequence elements simultaneously
  - Used in early speech recognition systems
  - Limited by fixed window size
  - Requires many input neurons
- **Recurrent Neural Networks (RNNs)**: Add recurrent connections to maintain state over time

12.2 Simple Recurrent Networks (SRNs)

SRNs (Elman networks) use feedback connections to process sequences:

- Architecture includes input, hidden, output, and context layers
- After each input, hidden layer activations are copied to context layer

- Context layer feeds back to hidden layer on the next time step
- Dynamics:

$$h_t = f(W_1 x_t + W_3 h_{t-1}) \tag{12.1}$$

$$o_t = g(W_2 h_t) \tag{12.2}$$

SRNs can be trained with backpropagation through time (BPTT), which unrolls the recurrent computation into a feedforward network across time steps.

12.3 Self-Supervised Learning

Self-supervised learning uses the data itself to generate supervision signals:

- In sequence learning, predict the next element given previous elements
- Allows learning from unlabeled sequential data
- Can discover structure in the data (e.g., Elman's SRN learned to cluster words by grammatical category)

12.4 Long Short-Term Memory (LSTM) Networks

Standard RNNs struggle with long-term dependencies due to vanishing gradients. LSTMs address this:

- Memory cells maintain information over long sequences
- Gating mechanisms control information flow:
  - Input gate: Controls what new information enters the cell
  - Forget gate: Controls what information is discarded
  - Output gate: Controls what information is used for output
- Successfully applied to speech recognition, machine translation, and text generation

12.5 Attention Mechanisms and Transformers

Attention mechanisms allow models to focus on relevant parts of input sequences:

- In sequence-to-sequence tasks, attention weights determine which input elements influence each output

- Self-attention allows elements to attend to other elements within the same sequence
- **Transformer Architecture**:
  - Relies entirely on self-attention, eliminating recurrence
  - Processes entire sequences in parallel rather than sequentially
  - Includes encoder (bidirectional context) and decoder (unidirectional context) components
  - Uses positional encoding to maintain sequence order
  - State-of-the-art for many NLP tasks

12.6 Other Self-Supervised Learning Approaches

Beyond sequence prediction, other self-supervised approaches include:

- **Contrastive Learning**: Learn representations by comparing similar vs. dissimilar examples
  - Maximize agreement between differently augmented views of the same data
  - Minimize agreement between views of different data
- **Masked Autoencoders**: Reconstruct images or text with portions masked out
  - Forces model to infer missing content from context
  - Effective pre-training strategy for both vision and language tasks

Chapter 13

Natural Language Processing

13.1 Evolution of NLP Methods

Natural language processing has evolved through several approaches:

- **Symbolic Methods** (1970s-1980s):
  - Hand-coded rules and dictionaries
  - Still used for preprocessing (e.g., tokenization)
- **Statistical Methods** (1990s-2000s):
  - Machine learning from large corpora
  - N-gram models and other statistical approaches
- **Neural Methods** (2010s-present):
  - Recurrent networks, then transformers
  - End-to-end learning without explicit feature engineering

13.2 Statistical Language Models

N-gram models predict the probability of a token based on preceding tokens:

- Unigram model: $P(w_1)P(w_2)...P(w_i)$
- Bigram model: $P(w_1)P(w_2|w_1)...P(w_i|w_{i-1})$
- Trigram model: $P(w_1)P(w_2|w_1)P(w_i|w_{i-2}, w_{i-1})$
- Parameters estimated by counting frequencies in training data
- Used to generate text by sampling from conditional distributions

13.3 Word Embeddings

Word embeddings represent words as dense vectors in a continuous space:

- Capture semantic relationships between words
- Words with similar meanings have similar vectors
- Popular methods:
  - Word2Vec: Predict context words (skip-gram) or target word (CBOW)
  - GloVe: Based on global word-word co-occurrence statistics
  - FastText: Incorporates subword information
- Properties:
  - Preserve semantic topology
  - Support limited compositionality through vector operations
  - Enable transfer learning in NLP tasks

13.4 Transformer-Based Models

Transformers have revolutionized NLP:

- Architecture based on self-attention without recurrence or convolution
- Two main components:
  - Encoder: Processes input bidirectionally (e.g., BERT)
  - Decoder: Generates output autoregressively (e.g., GPT)
- Key components:
  - Self-attention layers
  - Feed-forward networks
  - Layer normalization
  - Residual connections
  - Positional encoding

13.5 Large Language Models (LLMs)

Recent years have seen the rise of increasingly large transformer-based language models:

- Trained on vast amounts of text from the internet
- Key examples:
  - BERT: Bidirectional encoder for understanding
  - GPT series: Autoregressive decoders for generation
  - T5, BART: Encoder-decoder models for sequence-to-sequence tasks
- Training process:
  - Self-supervised pre-training on unlabeled text
  - Potential fine-tuning for specific tasks
  - Reinforcement Learning from Human Feedback (RLHF) for alignment
- Scaling laws: Performance improves predictably with model size, data, and compute
- Exhibit emergent abilities not present in smaller models

13.6 Text Generation and Decoding Methods

LLMs generate text by predicting the next token given previous tokens:

- **Decoding Strategies**:
  - Greedy decoding: Select most probable token
  - Top-k sampling: Sample from k most probable tokens
  - Top-p (nucleus) sampling: Sample from tokens comprising p probability mass
  - Temperature scaling: Control randomness by scaling logits
- **Chain-of-Thought Prompting**:
  - Providing step-by-step reasoning examples
  - Elicits more complex reasoning in LLMs
  - Shows emergent abilities in larger models

Chapter 14

Learning and Memory Systems

14.1 Types of Memory in Neural Networks

Neural networks implement different types of memory:

- **Long-Term Memory**: Connection weights represent stable knowledge
  - Modified gradually through learning
  - Captures statistical regularities
  - Analogous to semantic memory in humans
- **Short-Term Memory**: Neural activations represent temporary information
  - Maintained through recurrent connections
  - Analogous to working memory in humans
- **Episodic Memory**: Not inherent in standard neural networks
  - Requires additional mechanisms like memory-augmented neural networks
  - Stores specific experiences

14.2 Catastrophic Forgetting and Complementary Learning Systems

Neural networks struggle with sequential learning of multiple tasks:

- **Catastrophic Forgetting**: New learning overwrites previous knowledge
  - Example: In AB-AC associative learning, learning AC pairs disrupts AB memories
  - Much stronger in neural networks than in humans
- **Complementary Learning Systems Theory**:
  - Proposes two separate memory systems in the brain
  - Hippocampus: Fast learning, sparse representations, pattern separation
  - Neocortex: Slow learning, distributed representations, generalization
  - Hippocampal replay during sleep/rest helps consolidate memories in neocortex

14.3 Continual Learning in Neural Networks

Approaches to mitigate catastrophic forgetting:

- **Rehearsal**:
  - Interleave new data with old examples
  - Impractical for large datasets or privacy-sensitive data
- **Pseudo-Rehearsal**:
  - Generate synthetic examples of previous tasks using a generative model
  - No need to store original data
- **Similarity-Weighted Interleaved Learning (SWIL)**:
  - Only rehearse examples similar to current input
  - More efficient than full interleaved learning

14.4 Spontaneous Brain Activity and Generative Models

The brain shows significant activity even at rest:

- Only 10% of brain energy goes to sensory processing

- Spontaneous activity shows organized patterns (resting state networks)
- Hypothesis: Spontaneous activity reflects top-down dynamics of generative models
  - During tasks: expectations and attention
  - During rest: optimization of brain's generative model
- Examples in the brain:
  - Cortical resting state correlations
  - Hippocampal replay and preplay sequences

Chapter 15

Towards Artificial General Intelligence (AGI)

15.1 The Concept of AGI

Artificial General Intelligence refers to AI with human-level capabilities across a broad range of tasks:

- Longstanding goal of AI since its founding in 1956
- Explicit aim of major AI companies (OpenAI, DeepMind, Anthropic)
- Controversial regarding timeline: some see early signs ("sparks"), others expect decades, some are skeptical it will ever be achieved
- Often connected to the concept of "emergent abilities" in large-scale models

15.2 Defining and Assessing AGI

Various frameworks attempt to define and assess progress toward AGI:

- **The Turing Test**: AI indistinguishable from humans in conversation
- **Strong AI (Searle)**: System with "a mind in exactly the same sense human beings have minds"
- **Task-Based Definitions**: Performance on specified cognitive tasks
- **Performance and Generality Framework**:
  - Performance: Depth of capabilities compared to humans
  - Generality: Breadth of tasks achieving target performance
  - Defines progressive levels of AGI achievement

15.3 The Grounding Problem and Embodied Cognition

A key debate concerns whether language models can truly understand language:

- **The Grounding Problem**: Can meaning be derived purely from text, or must it be connected to sensorimotor experience?
- **Arguments Against LLM Understanding**:
  - LLMs are "stochastic parrots" trained on form, not meaning
  - Understanding requires connecting words to sensations and actions
  - Linguistic competence doesn't imply real-world understanding
- **Arguments For LLM Understanding**:
  - Performance improvements with scale suggest emerging comprehension
  - Similar processing patterns to human language understanding
  - Demonstrated capability to reason and solve novel problems
- **Embodied Cognition Theory**:
  - Cognition is shaped by bodily experiences and environment interaction
  - Concepts, even abstract ones, arise from sensorimotor foundations
  - Language processing activates sensorimotor regions in the brain

15.4 Scaling Laws and Emergent Abilities

Recent research suggests predictable patterns in AI development:

- **Scaling Laws**: Performance improves as power laws of:
  - Model parameters
  - Dataset size
  - Compute used for training
- **Emergent Abilities**:
  - Capabilities appearing suddenly in larger models that weren't present in smaller ones

  - Not predictable by extrapolating performance from smaller models
  - Examples: Chain-of-thought reasoning, instruction following, few-shot learning

15.5 Current Capabilities and Limitations

State-of-the-art models show impressive capabilities but notable limitations:

- **Capabilities**:
  - Common-sense reasoning and problem-solving
  - Visuo-spatial reasoning (even in language-only models)
  - Social reasoning about intentions and beliefs
  - Few-shot learning of new tasks
- **Limitations**:
  - Inconsistent numerical reasoning
  - Difficulty with certain visuo-spatial tasks
  - Hallucination of facts
  - Limited robustness to distribution shifts

Chapter 16

Perspectives and Challenges for Deep Learning AI

16.1 Robustness Challenges

Deep learning models face several robustness challenges:

- **Adversarial Examples**: Tiny, imperceptible perturbations that cause misclassification
- **Distribution Shift**: Performance degradation when test distribution differs from training
- **Out-of-Distribution Generalization**: Difficulty applying learned concepts to novel contexts
- **Context Dependence**: Failure to incorporate contextual information for interpretation

16.2 Explainable AI (XAI)

As AI systems make more consequential decisions, explainability becomes crucial:

- **Black Box Problem**: Deep networks are difficult to interpret
- **Dimensions of Explainability**:
  - Transparency: Explaining how a task is performed
  - Responsibility: Justifying choices and decisions
- **Approaches to XAI**:
  - Feature visualization and attribution methods

  - Attention visualization
  - Rule extraction
  - Contrastive explanations

16.3 AI Fairness and Ethics

AI systems can inherit or amplify societal biases:

- **Bias Sources**:
  - Training data reflecting historical inequalities
  - Algorithmic design choices
  - Evaluation metrics that don't consider fairness
- **Examples of Bias**:
  - Racial disparities in recidivism prediction
  - Gender stereotypes in language models
  - Demographic disparities in facial recognition
- **Ethical Considerations**:
  - Alignment to human values
  - Accountability for AI decisions
  - Transparency about AI capabilities and limitations
  - Privacy concerns

16.4 Regulatory Approaches

Governments are developing frameworks to address AI risks:

- **European Union's AI Act**:
  - Risk-based approach classifying AI systems
  - Prohibits systems posing unacceptable risk
  - Imposes requirements on high-risk systems
  - Transparency requirements for generative AI
- **Human-Centered AI**:
  - Emphasis on human control
  - Augmenting rather than replacing human capabilities
  - "Human-in-the-loop" approaches
  - Ethical and responsible development

16.5 Future Directions

Key areas for future development:

- **Multimodal Integration**: Combining language, vision, and other modalities
- **Embodied AI**: Grounding language in physical interaction
- **Causal Reasoning**: Moving beyond correlation to understanding causality
- **Continual Learning**: Adapting to new tasks without forgetting
- **Energy Efficiency**: Reducing the computational and environmental costs of AI

16.6 Conclusion

Deep learning has transformed AI, but significant challenges remain:

- Current technology excels in specific, circumscribed problems
- Explainability, robustness, and fairness are major limiting factors
- Human-centered approaches offer the best framework for responsible development
- Continued progress requires interdisciplinary collaboration between AI, cognitive science, neuroscience, and ethics

Bibliography

[1] Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[3] Koller, D., & Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.

[4] Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179-211.

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[6] Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. Science, 350(6266), 1332-1338.

[7] McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review, 102(3), 419.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[9] Marr, D. (1982). Vision: A computational investigation into the human representation and processing of visual information. MIT Press.

[10] Stoianov, I., & Zorzi, M. (2012). Emergence of a 'visual number sense' in hierarchical generative models. Nature neuroscience, 15(2), 194-196.

[11] Testolin, A., Stoianov, I., & Zorzi, M. (2017). Letter perception emerges from unsupervised deep learning and recycling of natural image features. Nature human behaviour, 1(9), 657-664.

[12] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. arXiv preprint arXiv:2206.07682.

[13] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

[14] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[15] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[17] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT Press.

[18] Pezzulo, G., Zorzi, M., & Corbetta, M. (2021). The secret life of predictive brains: what's spontaneous activity for? Trends in Cognitive Sciences, 25(9), 730-743.

[19] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., ... & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv preprint arXiv:2303.12712.

[20] Mitchell, M., & Krakauer, D. C. (2023). The debate over understanding in AI's large language models. Proceedings of the National Academy of Sciences, 120(13), e2215907120.