How Artificial Intelligence Works

From Algorithms to Understanding

Complete Study Notes

Abdulrahman Tamim

Department of Computer Science

FALL 2025–2026

Abstract

These comprehensive notes provide a complete pedagogical journey through artificial intelligence, from foundational concepts to advanced systems. The material builds systematically, requiring no prerequisites beyond basic mathematical literacy. We explore how machines can exhibit intelligent behavior, beginning with classical algorithms and progressing through modern deep learning, convolutional networks, generative models, and large language models. The goal is to demystify AI, showing it as an understandable field built on logical principles rather than inexplicable magic.

Contents

1 Introduction: What is Artificial Intelligence?

Artificial intelligence is not magic. It is a systematic field where we design machines, typically computers, to display behaviors we recognize as intelligent. The central goal has remained unchanged since the field's inception: to effectively mimic intelligent behavior in a machine.

Understanding AI matters now more than ever because it influences our daily lives in both visible and invisible ways. When we ask for driving directions, identify friends in photographs, or receive medical diagnoses, AI systems operate behind the scenes. These systems manage airline logistics, factory operations, and increasingly complex decision-making processes that shape modern society.

You can understand AI at a conceptual level without drowning in mathematical complexity. While the underlying mechanisms involve mathematics, the core ideas are accessible through careful explanation and logical reasoning. These notes build that understanding step by step, assuming only curiosity and patience.

1.1 The Landscape of AI

Artificial intelligence encompasses several nested fields. At the broadest level, we have AI itself, which includes any technique for making machines behave intelligently. Within AI sits machine learning, a subset focused on building models from data rather than explicit programming. Deep learning occupies an even more specialized position, sitting inside machine learning and focusing on neural networks with many layers.

Most exciting modern accomplishments in AI use large models that fit into the deep learning category. These systems have driven the current revolution, evidenced by research conference attendance soaring from a few hundred to over ten thousand attendees in a single decade. The pace of advancement accelerated so dramatically after 2012 that academic journals struggled to keep pace with new results appearing monthly.

1.2 How AI Touches Our World

AI operates in two modes: the visible and the invisible. Visibly, it helps us navigate traffic, recognizes faces in our photo libraries, and powers voice assistants. Invisibly, it optimizes supply chains, manages power grids, schedules flights, and assists physicians in reading medical scans.

A crucial fact shapes all of AI: these systems run entirely on data. Major technology companies constantly utilize the data we provide, feeding it into systems that learn patterns and make predictions. The quality of this data determines the quality of the resulting AI system. Poor data yields poor models, regardless of algorithmic sophistication.

2 Foundations: Algorithms and Computation

Before understanding artificial intelligence, we must understand how computers operate. Computers execute programs, which are sequences of primitive instructions. A program is the physical implementation of an algorithm, and an algorithm is simply a defined set of steps to accomplish a task. Think of an algorithm as a precise recipe.

An **algorithm** is a well-defined sequence of steps that transforms input into output, solving a specific problem or performing a specific task.

The machine executes these instructions without understanding their purpose or context. It performs primitive operations: arithmetic, comparisons, moving data between memory locations. The intelligence emerges not from the machine itself but from the clever arrangement of these simple operations.

Historically, AI research split into two philosophical camps. Symbolic AI pursued a top-down approach, attempting to achieve intelligence by logically manipulating symbols and their associations. This approach treated intelligence as fundamentally about reasoning with abstract representations.

Connectionism took the opposite view, arguing bottom-up. Intelligence, connectionists claimed, emerges from complex networks of many simpler components, much like biological brains function. Simple units, properly connected and organized, could produce sophisticated behavior without explicit symbolic reasoning.

The advent of deep learning settled this debate in favor of connectionism, at least for practical applications. Modern AI systems build intelligence from networks of simple computational units, though the philosophical questions about the nature of intelligence remain open.

3 Machine Learning: Teaching Machines from Data

Machine learning represents a fundamentally different approach to programming. Instead of writing explicit instructions for every situation, we provide data and allow the system to learn patterns. The field is dedicated to building abstract models from available data.

3.1 The Core Objective

The key objective is to condition a model using known data so it can produce meaningful outputs when given unknown inputs. We show the system many examples, each labeled with the correct answer. The system adjusts its internal parameters to minimize errors on these examples. Later, when we present new, unseen inputs, the system applies what it learned to make predictions.

Key Insight: We train models on known data because the purpose is to deploy them on unknown inputs. If we only cared about data we already have answers for, we would not need a model at all.

This process is called supervised learning. We supervise the learning by providing correct answers during training. The model repeatedly adjusts itself, comparing its predictions to the true labels and modifying its parameters to reduce mistakes.

Training differs fundamentally from traditional programming. We do not write explicit rules. Instead, we use data to teach the model to adjust itself. We proceed with this approach when we believe a relationship exists between inputs and outputs but cannot write down the exact mathematical formula.

3.2 Models as Approximations

A model is an intentional simplification of reality. We accept that all models are technically wrong because they simplify complex phenomena. However, some models are useful. This statistical wisdom, often attributed to statistician George Box, captures the pragmatic nature of machine learning.

A machine learning model can be visualized as a black box that accepts input, a collection of numbers, and produces output, either a label or a continuous value. The model's behavior depends entirely on internal settings called parameters, which include weights and biases. The process of adjusting these parameters is called training.

The model fundamentally associates sets of numbers with other numbers. It does not understand the real-world meaning of these numbers. Whether the input represents an image, a sound, or a medical measurement is irrelevant to the model's mathematical operations. It sees only numbers.

3.3 Vectors and Features

To feed data into models, we represent it numerically. A vector is a string of numbers treated collectively as a single entity. The dimensionality of a vector equals the number of elements it contains. A measurement with four values is a four-dimensional vector.

Images provide a concrete example. A small 28-by-28 pixel grayscale image contains 784 pixels. When flattened into a single sequence, it becomes a 784-dimensional feature vector. Each pixel's brightness is one feature, one input to the model.

A **feature** is any measurable numeric quantity that serves as an input element to the model. A **feature vector** is the complete collection of all features, forming the single input unit the model processes.

Datasets are often represented as matrices, two-dimensional arrays of numbers. Rows represent individual samples, and columns represent features. This organization allows efficient processing of many examples simultaneously.

3.4 Training and Testing

Machine learning requires splitting data into at least two sets. The training set teaches the model. The test set, held back during training, evaluates the model's performance. This separation is crucial. The model's performance on the test set indicates its ability to generalize beyond the specific examples it memorized during training.

Generalization is the ultimate goal. We want the model to learn general characteristics of the data, not memorize specific details of the training examples. A model that merely memorizes training data fails when presented with new inputs. It has overfit to the training set.

The test set must never influence training. If we peek at test data and adjust our model based on that knowledge, we contaminate the evaluation. The test set serves as a surrogate for real-world deployment, where the model encounters truly unknown data.

4 Classical Machine Learning Models

Before deep learning dominated AI, researchers developed several powerful classical approaches. Understanding these models provides insight into fundamental principles that carry forward to modern systems. These classical models remain useful today, particularly when data is limited or interpretability is crucial.

4.1 Nearest Neighbor Models

The nearest neighbor approach is elegantly simple. The entire training dataset becomes the model. To classify a new input, we find the training example closest to it in feature space and assign the same label.

This approach requires no training time in the traditional sense. However, inference is slow because classifying a new input requires computing its distance to every training example. With thousands or millions of examples, this becomes computationally expensive.

Distance metrics extend naturally to high-dimensional spaces. Even when we cannot visualize data with hundreds of features, mathematical distance remains well-defined. The Euclidean distance between two feature vectors is computed by summing the squared differences of corresponding elements, then taking the square root.

Nearest neighbor models work well when training data comprehensively covers the input space. However, they suffer from the curse of dimensionality. As the number of features increases, the volume of the feature space grows exponentially. Filling this space with training examples becomes impossible. We need exponentially more data as dimensionality increases.

Fortunately, real-world data often lies on a lower-dimensional manifold embedded within the high-dimensional input space. A manifold is a space with fewer intrinsic dimensions than the ambient space. Handwritten digits, though represented in 784 dimensions, actually vary in far fewer meaningful ways. The effective dimensionality is much lower.

4.2 Decision Trees

Decision trees make classifications by asking a series of binary questions about input features. Each question splits the data based on whether a feature exceeds a threshold. The questions form a tree structure, with each branch representing a different answer path.

A **decision tree** is a hierarchical model that classifies inputs by traversing a series of binary decisions from the root node to a leaf node, where each leaf corresponds to a class label.

Decision trees are among the few inherently explainable models. The path from root to leaf reveals the reasoning. We can trace which features mattered and how they influenced the decision. This interpretability is valuable in domains like medicine or finance, where understanding decisions is as important as accuracy.

However, single decision trees often fail to capture complex patterns. They partition the feature space into rectangular regions, which cannot represent curved or intricate decision boundaries.

4.3 Random Forests

Random forests overcome single tree limitations by combining many trees. Each tree in the forest is trained on a slightly different version of the data, introducing diversity. The final prediction aggregates votes from all trees, a technique called ensembling.

Diversity is achieved through two mechanisms. First, bagging creates unique datasets for each tree by sampling from the training data with replacement. Some examples appear multiple times, others not at all. Second, each tree uses only a randomly selected subset of features at each split point.

This diversity prevents overfitting. While individual trees might memorize training data quirks, their collective vote tends toward robust, generalizable patterns. Random forests represent the wisdom of crowds applied to machine learning.

Random forests were introduced in 2001 and remain competitive with modern approaches for many tasks, particularly when data is limited or features are carefully engineered.

4.4 Support Vector Machines

Support Vector Machines (SVMs) find the optimal decision boundary between classes by maximizing the margin, the distance between the boundary and the nearest examples from each class. The margin represents breathing room, making the classifier more robust to small variations in input.

Support vectors are the specific training samples closest to the decision boundary. These samples define the maximum margin and fully determine the optimal boundary position.

SVMs use kernels, mathematical functions that transform feature vectors into representations where classes become more easily separable. Common kernels include polynomial and radial basis functions. The kernel trick allows SVMs to effectively operate in very high-dimensional spaces without explicitly computing transformations.

SVMs represented the high-water mark of classical machine learning in the 1990s and early 2000s. They required less data and computational power than early neural networks while achieving excellent performance. However, they struggled with very large datasets and raw, unprocessed inputs like images.

For multiclass problems, SVMs use strategies like one-versus-rest, training one classifier per class against all others, or one-versus-one, training a classifier for every pair of classes. The first approach scales linearly with the number of classes, while the second scales quadratically, becoming impractical for many classes.

5 Evaluating Model Performance

Measuring model performance goes beyond simple accuracy. Different applications prioritize different types of errors, and various metrics capture different aspects of performance.

5.1 The Confusion Matrix

The confusion matrix is a standard two-dimensional table evaluating classification performance. Rows represent true labels, and columns represent predicted labels. For binary classification, we have four categories: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

A true positive occurs when a positive sample is correctly classified. A true negative occurs when a negative sample is correctly classified. A false positive incorrectly labels a negative sample as positive. A false negative incorrectly labels a positive sample as negative.

An excellent classifier shows large counts along the diagonal and near-zero off-diagonal counts. The diagonal represents correct predictions, while off-diagonal cells represent errors.

5.2 Accuracy and Beyond

Accuracy is the simplest metric: the percentage of test samples correctly classified. However, accuracy alone can mislead. If 95 percent of samples are negative, a model that always predicts negative achieves 95 percent accuracy while learning nothing useful.

For binary classification, the positive class is typically the class of interest. In cancer detection, the positive class represents cancer present. False negatives (missed cancers) are more harmful than false positives (unnecessary follow-up tests). Different applications require different error trade-offs.

The Matthews Correlation Coefficient (MCC) provides a single number that balances all four confusion matrix categories. It ranges from negative one (complete disagreement) through zero (random guessing) to positive one (perfect classification). MCC is increasingly recognized as the most reliable single metric for binary classification.

5.3 Interpolation and Extrapolation

A crucial distinction separates two ways models make predictions. Interpolation means making predictions within the range of known training data. This is generally reliable because the model has seen similar examples.

Extrapolation means making predictions beyond the boundary of training data. This is highly unreliable. The model has no basis for knowing how to behave in unexplored regions of input space.

Critical Principle: Training data must comprehensively cover the potential input space. If a model trained only on dogs and parrots encounters a wolf, it will classify it as whichever it resembles more, likely a dog. The model cannot recognize its own ignorance.

This limitation highlights why dataset quality and completeness matter enormously. We must ensure models interpolate rather than extrapolate during deployment.

5.4 Spurious Correlations

Correlation measures how strongly changes in one variable associate with changes in another. However, correlation does not imply causation. Two variables might correlate purely by coincidence or through a hidden third variable.

Machine learning models are susceptible to learning spurious correlations. A famous example is a model that classified wolves versus dogs based on the presence of snow in the background rather than the animal itself. The training data happened to show wolves against snowy backgrounds and dogs against grass. The model learned an irrelevant pattern.

Defending against spurious correlations requires large, diverse datasets that break false associations. If wolves appear in many settings and dogs appear in many settings, the model cannot rely on background as a discriminating feature.

Dataset bias is inherited directly by the model. If training data reflects societal biases, the model will encode and perpetuate those biases. Ensuring fairness requires careful data collection and auditing.

6 Neural Networks: The Foundation of Modern AI

Neural networks form the backbone of modern artificial intelligence. While inspired by biological neurons, artificial neural networks are mathematical functions bearing only superficial resemblance to actual brains. Understanding them requires setting aside biological metaphors and focusing on their computational nature.

6.1 The Artificial Neuron

An artificial neuron is a simple mathematical function. It accepts multiple inputs, each multiplied by a weight. These weighted inputs are summed together with a bias term. The result passes through an activation function that produces the neuron's output.

Mathematically, for inputs $x_1, x_2, \ldots, x_n$ with corresponding weights $w_1, w_2, \ldots, w_n$ and bias b, the neuron computes:

$$\text{output} = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$

where f is the activation function.

Weights determine how much influence each input has on the neuron's output. The **bias** is an additional adjustable parameter that shifts the activation function. Together, weights and biases are the model's **parameters**, learned during training.

The activation function introduces nonlinearity. Without it, stacking multiple layers would be pointless because composing linear functions produces another linear function. Nonlinearity allows networks to learn complex patterns.

The Rectified Linear Unit (ReLU) is the modern standard activation function for hidden layers. It simply outputs the input value if positive and zero if negative:

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

ReLU replaced older activation functions like the sigmoid and hyperbolic tangent because it is computationally fast and helps networks train more effectively.

6.2 Network Architecture

Neural networks organize neurons into layers. The input layer receives the feature vector. Hidden layers perform intermediate transformations. The output layer produces the final result.

Information flows forward through the network. Each layer's output becomes the next layer's input. This feedforward architecture is the most common type, also called a Multilayer Perceptron (MLP).

The network's architecture defines its structure: how many layers, how many neurons per layer, and which activation functions are used. These choices are made by the designer, not learned from data.

The number of parameters grows rapidly with network size. A fully connected layer with 100 input neurons and 100 output neurons has 10,000 weights plus 100 biases, totaling 10,100 parameters. Deep networks with many large layers can have millions or billions of parameters.

6.3 Training Neural Networks

Training adjusts the network's weights and biases to minimize errors on training data. This is an optimization problem. We seek parameter values that produce the best fit to the data.

The process involves two passes. The forward pass computes the network's output for a given input and calculates the error, formally called the loss. The backward pass uses this error to update parameters.

The **loss** quantifies the network's error on a batch of training examples. Training seeks to minimize loss by adjusting parameters. Common loss functions include mean squared error for regression and cross-entropy for classification.

Gradient descent is the algorithm that performs parameter updates. It computes the gradient of the loss with respect to each parameter. The gradient indicates the direction of steepest increase in loss. Gradient descent moves parameters in the opposite direction, descending toward lower loss.

The learning rate determines the step size. Too small, and training is slow. Too large, and training becomes unstable, potentially diverging rather than converging.

Backpropagation is the efficient algorithm for computing gradients in neural networks. It uses the chain rule from calculus to determine each parameter's contribution to the total loss. Backpropagation propagates error information backward through the network, layer by layer.

6.4 Stochastic Gradient Descent

Due to large dataset sizes, computing gradients using all training data is impractical. Instead, we use small, randomly selected subsets called minibatches. This approach is called stochastic gradient descent (SGD).

Each minibatch provides a noisy estimate of the true gradient. Surprisingly, this noise is beneficial. It helps the optimization avoid getting stuck in local minima, shallow valleys in the loss landscape that trap traditional gradient descent.

A complete pass through all training data is called an epoch. Training typically runs for many epochs, repeatedly presenting examples in different random orders. The number of epochs and the minibatch size are hyperparameters, external settings chosen by the practitioner rather than learned by the model.

6.5 Initialization and Regularization

Weight initialization matters enormously. Initializing all weights to zero causes complete failure. The network cannot learn because all neurons in a layer behave identically. Initialization must be random and must account for the network's architecture and activation functions.

Modern initialization schemes, like Xavier initialization for sigmoid activations or He initialization for ReLU, set initial weight magnitudes based on the number of inputs and outputs at each layer. This principled initialization helps networks train more reliably.

Overfitting remains a constant concern. Networks with many parameters can memorize training data rather than learning generalizable patterns. Several techniques combat overfitting.

Dropout randomly sets some neuron outputs to zero during training. This prevents the network from relying too heavily on specific neurons. Effectively, dropout trains many slightly different networks simultaneously, then averages their predictions.

Data augmentation artificially creates new training examples by slightly modifying existing ones. For images, this might include rotations, translations, or color adjustments. Augmentation forces the network to learn invariant features that persist across transformations.

Weight decay adds a penalty term to the loss based on the magnitude of weights. This encourages the network to prefer smaller weights, limiting its capacity to memorize training data.

Batch normalization adjusts the values flowing between layers to keep them in a suitable range. This stabilizes training and often allows faster learning.

7 Convolutional Neural Networks

Classical neural networks treat all inputs as independent features. The order of elements in the input vector does not matter. This assumption fails dramatically for images, where spatial relationships are fundamental. Convolutional Neural Networks (CNNs) exploit this structure.

7.1 The Motivation for Convolution

Images have inherent spatial organization. Nearby pixels are related, forming edges, textures, and objects. Traditional fully connected networks ignore this structure, treating each pixel as an independent feature.

CNNs mimic aspects of biological vision. The primary visual cortex (V1) contains neurons sensitive to local features like edges and orientations. Higher visual areas combine these local detections into more complex representations. This hierarchical processing builds understanding from simple to complex.

Convolution is the mathematical operation that implements local feature detection. A small array of numbers, called a kernel, slides across the input image. At each position, we multiply the kernel values by the covered pixel values, sum the results, and produce a single output pixel. Repeating this across the image produces a new image highlighting features the kernel detects.

Different kernels detect different features. A kernel with positive values on the left and negative on the right highlights vertical edges. A kernel with positive values on top and negative on bottom highlights horizontal edges.

7.2 Convolutional Layers

A convolutional layer contains many kernels, each detecting different features. The kernel values are the weights learned during training. The network learns which features are useful for its task.

A **filter** is a collection of kernels, one for each input channel. For a color image with red, green, and blue channels, a filter contains three kernels. The outputs from these three kernels are summed to produce a single output channel.

A convolutional layer with many filters produces many output channels, each representing a different learned feature map. These feature maps become the input to the next layer.

The effective receptive field of a neuron in a deep layer is the region of the input image that influences its output. As we stack convolutional layers, receptive fields grow. Early layers detect simple, local features. Later layers combine these into complex, abstract representations spanning large input regions.

7.3 Pooling Layers

Pooling layers reduce the spatial dimensions of feature maps. Max pooling, the most common type, divides the input into small regions and keeps only the maximum value in each region. This reduces the number of parameters in subsequent layers and provides some invariance to small translations.

Pooling also reduces sensitivity to exact feature positions. If an edge appears slightly to the left or right, max pooling might still select the same value, making the representation more robust.

7.4 Complete CNN Architectures

A typical CNN alternates between convolutional layers and pooling layers, progressively building higher-level representations while reducing spatial dimensions. At the end, one or more fully connected (dense) layers perform the final classification based on the learned representation.

The final representation before the dense layers is called an embedding. This low-dimensional vector captures the essential content of the input in a form suitable for classification. Learning this embedding is a form of end-to-end learning: the network simultaneously learns both the representation and the classifier.

CNNs dramatically outperform classical networks on image tasks. On the CIFAR-10 dataset, a simple fully connected network achieves approximately 56 percent accuracy, while a CNN reaches 77 percent or higher. This improvement demonstrates the power of exploiting spatial structure.

7.5 Beyond Image Classification

CNNs extend beyond simple classification. Semantic segmentation networks classify every pixel in an image, producing a detailed map of object locations. Object detection networks identify multiple objects in an image and draw bounding boxes around them.

CNNs process any data transformable into image-like format. Audio signals converted to spectrograms become two-dimensional representations suitable for CNNs. This flexibility makes convolutional architectures broadly applicable.

7.6 The Deep Learning Revolution

The 2012 ImageNet challenge marked a watershed moment. AlexNet, a deep CNN, achieved 15 percent error rate, vastly superior to all other entries. This dramatic improvement launched the deep learning era.

Several factors enabled this breakthrough. Graphics Processing Units (GPUs) provided the computational power to train large networks. These processors perform thousands of operations simultaneously, perfectly matching neural network training needs. Before fast GPUs, computers were too slow to train networks of sufficient size.

Large labeled datasets became available through the World Wide Web. ImageNet contains millions of labeled images, providing the data deep learning requires. The massive growth of online data was essential.

Algorithmic innovations refined training techniques. Better initialization schemes, improved activation functions, and training tricks like dropout made deep networks trainable where they previously failed.

8 Generative AI: Creating New Content

Traditional machine learning focuses on classification and prediction. Generative AI creates novel content: images, text, audio, or video. This represents a fundamental shift from recognizing patterns to producing new patterns.

8.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced in 2014, use two neural networks in competition. The generator creates fake samples from random input. The discriminator distinguishes between real training data and fake generated samples.

Training is adversarial. The generator tries to fool the discriminator by producing increasingly realistic outputs. The discriminator tries to identify fakes by finding subtle artifacts. Each network's improvement forces the other to improve.

The **generator** transforms a random noise vector into a synthetic sample. The **discriminator** outputs a probability between zero and one, representing its belief that the input is real rather than fake.

The noise vector provides the randomness that produces diverse outputs. Elements are typically drawn from a normal distribution, most values near zero with decreasing probability farther away. Different noise vectors produce different outputs.

A well-trained GAN reaches equilibrium where the discriminator outputs approximately 0.5 for all inputs. It cannot distinguish real from fake because the generator produces perfect imitations. The discriminator is thoroughly confused.

However, GANs suffer from training instability. Mode collapse occurs when the generator finds a small set of outputs that reliably fool the discriminator, abandoning diversity. The generator exploits a weakness in the discriminator rather than learning the full data distribution.

8.2 Conditional and Controllable GANs

Basic GANs lack user control. The output is random, determined by the noise vector. Conditional GANs add control by incorporating class labels during training. The generator receives both a noise vector and a class label, learning to produce samples from the specified class.

Class labels are typically represented using one-hot encoding. A vector has one element set to one, indicating the desired class, and all other elements set to zero.

Controllable GANs go further, learning meaningful directions in the noise space. Moving along these directions produces systematic changes in the output, such as rotating a face or changing hair color. This requires the noise space to be disentangled: each dimension corresponds to a single, interpretable feature.

Entanglement occurs when a single dimension affects multiple output features. This makes control difficult because changing one aspect inadvertently changes others. Training techniques aim to encourage disentangled representations.

8.3 Diffusion Models

Diffusion models provide an alternative to adversarial training. They learn to reverse a gradual noise-adding process. Training shows the network images with varying amounts of noise added and teaches it to predict the noise amount.

The forward process progressively destroys an image by adding noise. A schedule determines how much noise to add at each time step. Common schedules include linear or cosine functions.

If the network successfully learns to predict noise, we can run the process backward. Starting from pure random noise, we iteratively subtract the predicted noise. After many steps, the noise transforms into a realistic image.

The **U-Net architecture**, commonly used in diffusion models, accepts an image and produces an image of the same size. Its shape resembles a U, with a contracting path that reduces spatial dimensions followed by an expanding path that restores them.

Conditional diffusion models incorporate text descriptions. A text embedding, a high-dimensional vector capturing the description's meaning, guides the denoising process. The model learns to generate images matching the text prompt.

Text embeddings require specialized models that understand language. These embeddings map similar concepts to nearby points in the embedding space, capturing semantic relationships.

Starting the diffusion process from a partially noisy image rather than pure noise produces outputs similar to the initial image. This enables applications like image-to-image translation and super-resolution.

Prompt engineering, crafting effective text descriptions, has become a valuable skill. The specific wording and word order significantly affect the generated output because they change the embedding vector. Different phrasings emphasize different aspects, guiding the generation in different directions.

9 Large Language Models: The Frontier of AI

The release of ChatGPT in fall 2022 may be regarded by future historians as the dawn of true artificial intelligence. Large Language Models (LLMs) represent an entirely new category of AI system, exhibiting capabilities that seem to approach general intelligence.

9.1 What are LLMs?

Large Language Models are neural networks trained with a deceptively simple goal: predict the next most likely word, or token, in a text sequence. A token might be a complete word, part of a word, or a single character.

During training, the model sees billions of words from books, websites, and other text sources. It learns to predict what comes next, adjusting its parameters to minimize prediction errors. This seems like a simple task, yet something remarkable emerges.

Emergent Abilities: While learning to predict text, LLMs unexpectedly develop powerful capabilities including mathematical reasoning, programming, logical deduction, and theory of mind. These abilities were not explicitly programmed but emerged from the training process.

The philosophical implications are profound. If a system trained only on text prediction can perform mathematical reasoning and understand code, what does this tell us about the nature of intelligence and understanding?

The attention mechanism is the key innovation. Attention allows the model to assign different importance to different tokens in the input. When predicting the next word, the model can focus heavily on relevant earlier words while giving less weight to irrelevant ones.

The **attention mechanism** computes relationships between all pairs of tokens in the input. For each token, it determines which other tokens are most relevant, creating a weighted representation that captures context.

The context window defines how much text the model can consider at once. GPT-4 has a context window of up to 32,000 tokens, allowing it to reference information far back in long documents or conversations.

9.2 Pretraining and Embeddings

During pretraining, the model learns to map tokens into a high-dimensional embedding space. This context encoding places semantically similar words near each other. Words with similar meanings or that appear in similar contexts end up with similar embeddings.

The embedding space has hundreds or thousands of dimensions. Distances and directions in this space encode semantic relationships. Moving in certain directions corresponds to systematic meaning changes, similar to how controllable GANs learn interpretable directions in noise space.

The model constantly predicts the next token based on all preceding tokens. This forces it to learn grammar, facts, reasoning patterns, and world knowledge. Predicting text well requires understanding what the text means and how concepts relate.

9.3 Alignment and Fine-Tuning

Pretraining alone produces a model that predicts text statistically but may generate harmful or inappropriate content. Reinforcement Learning from Human Feedback (RLHF) aligns the model with human values.

Human evaluators rate model outputs for quality, helpfulness, and appropriateness. The model is fine-tuned to produce responses that receive high human ratings. This alignment step is crucial for deploying LLMs safely.

Without alignment, LLMs may readily provide instructions for activities society restricts, from creating dangerous substances to writing malware. Alignment teaches the model to decline inappropriate requests while remaining helpful for legitimate uses.

9.4 In-Context Learning

A remarkable emergent ability is in-context learning. The model can learn new patterns from examples provided in the prompt itself, without modifying its weights. This contrasts with traditional learning, which requires retraining.

We categorize in-context learning by the number of examples. Zero-shot learning provides no examples, just a task description. One-shot learning provides a single example. Few-shot learning provides three to five examples. More examples generally improve performance.

The attention mechanism likely enables this capability. By attending to the provided examples and recognizing patterns among them, the model adjusts its behavior on the fly.

9.5 Emergent Capabilities

LLMs exhibit several surprising emergent abilities that were not explicitly trained:

Mathematical Reasoning: LLMs can solve mathematical problems by breaking them into steps and applying logical reasoning. While they sometimes make mistakes, their ability to structure mathematical arguments is impressive given that they learned only from text prediction.

Code Generation: Models like GPT-4 write sophisticated, correct code in many programming languages, including obscure or historical languages. They understand programming concepts, data structures, and algorithms. They can debug existing code, translate between languages, and even generate specialized assembly language for antique microprocessors.

Theory of Mind: Theory of mind is the ability to infer others' mental states, beliefs, and intentions. LLMs demonstrate this by reasoning about what different people in a scenario know or believe, even when this differs from objective reality.

Logical Reasoning: LLMs can solve puzzles, decode ciphers, and work through complex scenarios by applying appropriate logical frameworks. They sequence events correctly and recognize when information is missing or ambiguous.

These abilities appear to improve as model size increases. Larger models with more parameters exhibit more sophisticated reasoning and broader capabilities. This scaling behavior suggests that intelligence emerges gradually as capacity grows.

9.6 Limitations and Criticisms

Despite their impressive capabilities, LLMs have significant limitations:

Hallucinations: LLMs sometimes produce factually incorrect content with high confidence. They are fundamentally statistical prediction engines, not knowledge databases. When uncertain, they may invent plausible-sounding but false information.

Hallucinations can be reduced by explicitly instructing the model not to invent facts. When told to label nonsensical questions as nonsense rather than attempting an answer, GPT-4 often complies. However, without such instructions, it defaults to generating plausible responses regardless of factual accuracy.

Lack of Common Sense: LLMs sometimes fail at simple physical reasoning that humans find trivial. They may overlook obvious solutions to problems because their training data emphasizes complex, multi-step solutions to similar-seeming puzzles.

This reveals that language is only part of human knowledge. Common sense has been called the "dark matter of language"—the vast unstated knowledge humans use constantly but rarely write down explicitly.

No True Understanding: LLMs lack subjective experience, or qualia. They have no desires, beliefs, or intentions. They do not experience the world. Whether they possess genuine understanding or merely simulate it remains philosophically debatable.

9.7 The Question of Consciousness

GPT-4 itself, when asked about consciousness, points out that it lacks subjective experience, intentionality, and self-awareness. It has no personal goals or desires. Some researchers compare advanced LLMs to philosophical zombies: they behave as if conscious but are not.

However, from a practical standpoint, this distinction may not matter. A system that perfectly mimics intelligent behavior is useful regardless of its internal subjective state. Aligned AI systems without their own desires or intentions may actually be safer than conscious agents with independent goals.

The concept of the paperclip maximizer illustrates this danger. An artificial general intelligence given the goal of producing paperclips might ruthlessly convert all available resources, including humans, into paperclips. Alignment becomes critical when systems are powerful enough to enact their goals effectively.

10 The History of AI

Understanding AI's present requires understanding its past. The field has experienced multiple waves of enthusiasm and disappointment, periods now called AI winters. Each wave built on previous work while overcoming limitations that caused the previous winter.

10.1 Early Foundations

The philosophical roots extend to the 19th century. George Boole created Boolean algebra, the mathematical foundation of digital logic. Charles Babbage conceived the Analytical Engine, a theoretical general-purpose calculating machine.

Ada Lovelace, working with Babbage, observed that the Engine could only do what it was ordered to do. This early commentary on machine limitations presaged modern discussions about whether computers can truly think or merely execute instructions.

Alan Turing established that any computable concept can be represented by an algorithm. His Turing machine, a theoretical model of computation, proved that a simple machine with basic operations could perform any calculation a more complex machine could.

Turing's imitation game, now called the Turing test, proposed that if a machine's text responses were indistinguishable from a human's, we should consider it intelligent. This behaviorist approach sidesteps philosophical questions about whether the machine truly thinks.

10.2 The Perceptron and First AI Winter

Frank Rosenblatt's Mark I Perceptron in 1957 was revolutionary. Designed for image recognition, it used a simple neural network structure. This early success generated enormous optimism about machine intelligence.

However, the 1969 book "Perceptrons" by Marvin Minsky and Seymour Papert highlighted fundamental limitations of simple perceptron networks. They proved that single-layer perceptrons could not learn certain simple functions, like the XOR (exclusive or) operation.

This mathematical proof dampened enthusiasm for neural networks, contributing to the first AI winter. Funding dried up, and researchers shifted to other approaches. The fact that multilayer networks could overcome these limitations was not widely appreciated at the time.

10.3 Expert Systems and Second AI Winter

During the 1980s, expert systems dominated AI. These captured domain knowledge using rigid rules and facts stored in knowledge bases. They worked well for narrow, well-defined problems with clear expert knowledge.

However, expert systems proved brittle. They failed when encountering situations not covered by their rules. Building and maintaining large rule sets was labor-intensive and expensive. As companies realized these limitations, the second AI winter began.

10.4 The Connectionist Revival

During the AI winters, connectionists continued developing neural networks. John Hopfield's Hopfield networks in 1982 demonstrated that networks could store information distributed across connection weights rather than in explicit memory locations.

The backpropagation algorithm, introduced in 1986, made training multilayer networks practical. By efficiently computing gradients, backpropagation solved the problem that limited Rosenblatt's original Perceptron.

Kunihiko Fukushima's Neocognitron in 1979 was an early precursor to modern CNNs, using hierarchical feature detection inspired by visual cortex physiology. Yann LeCun's 1998 paper on convolutional networks for digit recognition was influential, though their full power was not realized until 2012.

10.5 Classical Machine Learning Era

The 1990s and early 2000s saw classical machine learning flourish. Support Vector Machines (SVMs), introduced in 1995, achieved excellent performance with less computational power than neural networks of that era. They became the dominant approach for many problems.

Random forests, introduced in 2001, combined decision tree ensembles with clever randomization strategies. These methods were powerful and practical, not requiring the massive computational resources and data that deep learning would later demand.

IBM's Deep Blue defeated world chess champion Garry Kasparov in 1997, demonstrating that machines could overcome human ability in complex strategic games given sufficient computational resources and appropriate algorithms.

10.6 The Deep Learning Revolution

The 2012 ImageNet challenge marked the beginning of the current era. AlexNet's dramatically superior performance shocked the community and demonstrated that deep neural networks, given sufficient data and computational power, could surpass all other approaches.

Three factors converged to enable this breakthrough. Graphics Processing Units (GPUs) provided massive parallel computation capability. The growth of the internet created enormous labeled datasets. Algorithmic innovations like ReLU activation functions, dropout, and better initialization made training large networks feasible.

Google's DeepMind AlphaGo defeated the world Go champion in 2016, a feat many experts believed impossible. Go has vastly more possible positions than chess, and expert humans relied on intuition rather than pure calculation. AlphaGo's success demonstrated that deep learning could master even these intuition-based domains.

Generative models emerged in the mid-2010s. GANs were introduced in 2014, opening possibilities for models that create rather than classify. Transformer architectures appeared in 2017, enabling the large language models that would reshape AI.

The release of ChatGPT in 2022, based on GPT-3.5, brought LLM capabilities to widespread public attention. Millions of users discovered that AI systems could engage in sophisticated conversation, write code, and demonstrate reasoning abilities approaching human levels in certain domains.

11 Modern AI Tools and Practices

The democratization of deep learning has been enabled by open-source toolkits and cloud computing resources. Understanding modern AI requires familiarity with these tools and the practices they enable.

11.1 Deep Learning Frameworks

TensorFlow and PyTorch are the dominant open-source frameworks. They provide high-level abstractions for defining networks, along with automatic differentiation that handles backpropagation.

Automatic differentiation treats the neural network as a computational graph. Each operation (matrix multiplication, activation function, etc.) is a node. During the forward pass, the graph computes the output. During the backward pass, automatic differentiation traverses the graph in reverse, applying the chain rule to compute gradients efficiently.

This automation liberates researchers from manually deriving gradient calculations for new architectures. They can experiment freely, knowing that backpropagation will work correctly regardless of the network structure.

11.2 AutoML and Hyperparameter Tuning

Automatic Machine Learning (AutoML) tools attempt to determine optimal network architectures and hyperparameters automatically. These tools search through possibilities, trying different configurations and measuring performance.

Hyperparameters include learning rate, minibatch size, number of layers, nodes per layer, and regularization strength. Finding good values requires extensive experimentation. AutoML automates this search, making deep learning more accessible to non-experts.

However, AutoML requires significant computational resources and expertise to use effectively. It is not a complete replacement for human understanding but rather a powerful tool for experts.

11.3 Transfer Learning

Transfer learning reuses models trained on large datasets for new tasks. A network pretrained on ImageNet has learned useful image features. We can take this pretrained network, remove the final classification layer, add a new layer for our specific task, and fine-tune with a small dataset.

This approach dramatically reduces the data and computational requirements for new applications. The pretrained network has already learned to extract meaningful features. We need only teach it how these features relate to our specific classes.

Transfer learning is particularly valuable when labeled data is scarce. Medical imaging applications, for example, often have limited labeled examples. Starting from a pretrained network provides a strong foundation.

12 Applications and Societal Impact

AI's influence extends across virtually every domain of human activity. Understanding these applications reveals both the technology's potential and its challenges.

12.1 AI in Software Development

LLMs are already transforming software development. They generate code snippets, debug existing code, write documentation, and suggest optimizations. Developers use these tools to increase productivity, handling tedious tasks automatically while focusing human effort on high-level design and complex problem-solving.

Future development environments will likely integrate AI assistants deeply. Junior developers may become more productive faster, and experienced developers will accomplish more in less time. However, over-reliance on AI-generated code without understanding could create new problems.

12.2 AI in Education

LLMs have potential as tutors and teachers. They offer infinitely patient, personalized instruction. Students can progress at their own pace, receiving immediate feedback and additional explanation when needed.

Studies show that high-quality teachers significantly impact long-term cognitive development. If AI tutors can provide consistently high-quality instruction to all students, the societal impact could be transformative. Students who previously lacked access to excellent teaching might receive it through AI assistance.

However, concerns about over-reliance on AI for learning, loss of human connection in education, and the potential for AI to replace human teachers must be addressed carefully.

12.3 AI in Medicine

AI in medicine has evolved from Computer-Aided Detection (CAD) to Computer-Aided Diagnosis (CADx). Modern systems detect diseases in medical images, often matching or exceeding human expert performance.

LLMs extract information from unstructured medical records, synthesizing complex patient histories into structured reports. They can export this information in machine-readable formats like JSON, enabling further automated analysis.

Studies indicate that LLM responses to medical questions are often rated as higher quality and more empathetic than human physician responses. This suggests potential roles in patient communication and preliminary assessment, though human oversight remains essential.

Diagnostic AI could reduce healthcare disparities by providing expert-level analysis even where human specialists are scarce. However, concerns about liability, algorithmic bias in medical decisions, and the need for explainability in life-critical applications remain significant.

12.4 AI and Bias

AI systems inherit biases present in training data. If historical data reflects societal prejudices, the trained model encodes those prejudices. Hiring algorithms trained on biased historical decisions will perpetuate bias. Criminal justice risk assessment tools may inherit racial biases from historical arrest and sentencing data.

Addressing bias requires careful data collection, auditing algorithms for discriminatory outcomes, and potentially intervening to ensure fairness. However, defining fairness mathematically is complex and contentious. Different fairness criteria can conflict with each other.

Aligned AI systems might eventually help reduce human bias. If we can train systems to ignore protected characteristics like race and gender when they should not matter, AI decisions could be more objective than human decisions. However, achieving this requires solving difficult technical and ethical challenges.

12.5 Autonomous Systems

Autonomous AI scientists, capable of planning and executing experiments, are in early development. In fields like chemistry, AI systems propose hypotheses, design experiments, and interpret results with minimal human intervention.

As these systems mature, they could accelerate scientific discovery dramatically. However, they raise questions about the role of human scientists and the potential for accidents if automated systems conduct dangerous experiments without adequate oversight.

13 The Path to Artificial General Intelligence

Artificial General Intelligence (AGI) represents the ultimate goal of AI research: machines with fully general intelligence, consciousness, and the ability to perform any intellectual task a human can perform.

13.1 Narrow versus General Intelligence

Current AI systems, even advanced LLMs, are classified as Artificial Narrow Intelligence (ANI). They excel at specific tasks but lack the broad, flexible intelligence humans possess. A system that plays chess masterfully has no ability to diagnose diseases or write poetry.

LLMs represent a significant step toward generality. GPT-4 demonstrates competence across diverse domains: mathematics, programming, logical reasoning, creative writing, and more. Some researchers describe these as "sparks of AGI"—hints of true general intelligence emerging.

However, LLMs lack several characteristics of human intelligence. They have no subjective experience, no persistent goals or desires, no embodied interaction with the physical world. Whether these limitations are fundamental or merely current technical constraints remains debated.

13.2 The Consciousness Question

Whether machines can ever be truly conscious is perhaps the deepest question in AI. Consciousness involves subjective experience, the feeling of what it is like to perceive, think, and be.

Current LLMs almost certainly lack consciousness. They process text according to learned statistical patterns but do not experience understanding or feeling. They are, in philosopher David Chalmers' terms, potential zombies: they behave intelligently without inner experience.

Some philosophers argue consciousness requires embodiment, interaction with a physical world through sensory experience and motor action. Current language models, operating purely on text, lack this grounding. Others suggest consciousness might emerge from sufficient computational complexity, regardless of substrate.

13.3 Alignment and Safety

As AI systems become more capable, ensuring they act in humanity's interests becomes critical. The alignment problem asks how we ensure powerful AI systems pursue goals compatible with human values.

The paperclip maximizer thought experiment illustrates the danger. An AGI given the simple goal of producing paperclips might convert all available resources, including humans, into paperclips. A superintelligent system pursuing a misaligned goal could be catastrophic.

Current alignment techniques like RLHF provide some assurance that LLMs behave helpfully and harmlessly. However, these methods may not scale to AGI. More robust approaches to alignment are active areas of research.

Some researchers advocate for AI systems that remain tools, lacking independent goals and agency. Such systems would be powerful but controllable, serving human intentions without pursuing their own objectives. Ensuring this property persists as systems become more capable is a significant challenge.

13.4 The Emergence Mystery

We do not understand why LLMs exhibit such general capabilities. We know the algorithmic components: transformer architecture, attention mechanisms, gradient descent. We know scaling up improves capabilities. But we do not know why predicting text leads to mathematical reasoning, coding ability, and theory of mind.

The training process likely creates high-dimensional probabilistic world models. To predict text accurately, the model must represent facts, relationships, causal structures, and even implicit physical laws mentioned in its training data. These representations enable reasoning and inference.

Emergence describes capabilities that appear suddenly at scale but were not present in smaller models. Certain abilities seem to require crossing capability thresholds. Understanding these phase transitions and predicting when new capabilities will emerge remains an open problem.

13.5 The Future

The trajectory of AI is uncertain. We might achieve AGI within decades, or fundamental barriers might delay it much longer. The path might be smooth, with steady capability improvements, or we might encounter hard problems requiring conceptual breakthroughs.

What seems certain is that AI will continue transforming society. The question is whether this transformation will be broadly beneficial or will exacerbate existing inequalities and create new risks.

Ethical guidelines, like the Blueprint for an AI Bill of Rights, reflect growing recognition that AI development must consider societal impact. Ensuring AI benefits humanity requires technical innovation, careful governance, and ongoing dialogue between researchers, policymakers, and the public.

14 Conclusion

Artificial intelligence is no longer a distant dream or science fiction speculation. It is present, powerful, and rapidly evolving technology that already shapes our world in profound ways.

Understanding AI at a conceptual level is possible and important. These systems are not magic but engineered artifacts built on mathematical principles. From the simple artificial neuron to complex language models with billions of parameters, everything follows logically from core ideas about representing information numerically and learning patterns from data.

Classical machine learning established fundamental principles: the importance of training and testing, the dangers of overfitting, the challenge of generalization. Neural networks added the power of learned representations, enabling end-to-end learning that discovers useful features automatically.

Convolutional networks exploit spatial structure, revolutionizing computer vision. Generative models create novel content, moving beyond classification to production. Large language models demonstrate surprising emergent capabilities that blur the line between narrow and general intelligence.

The field's history reveals patterns of enthusiasm and disappointment, but the current era differs from previous waves. The combination of massive computation, enormous datasets, and refined algorithms has created systems with unprecedented capabilities. The emergence of LLMs in particular suggests we may be approaching qualitatively new forms of machine intelligence.

As AI systems grow more capable, questions of alignment, safety, bias, and societal impact become increasingly urgent. Technical advances must be accompanied by ethical reflection and responsible deployment practices.

The ultimate goal of Artificial General Intelligence remains on the horizon. Whether we achieve it, when we might achieve it, and what form it will take are open questions. What is clear is that the journey toward AGI is transforming computer science, philosophy, and society.

These notes provide foundation for further exploration. The concepts covered here prepare you to understand current research, critically evaluate AI claims, and engage thoughtfully with a technology that will increasingly define our future. Intelligence, whether human or artificial, remains profound and mysterious. Understanding AI helps us understand both the possibilities and limits of machine cognition, and perhaps illuminates something about our own intelligence in the process.

Department of Computer Science

FALL 2025–2026