From Neurons to Foundation Models: A Systematic Review of Convolutional Networks, Recurrent Architectures, Attention Mechanisms, the Transformer Paradigm, and the Era of Generative Artificial Intelligence

Author

Asrat Amnie, Associate Professor

Affiliation

The City University of New York

February 22, 2026

(Unpublished Work)

Abstract

The past three decades have produced the most consequential transformation in the history of artificial intelligence, driven by a cascade of architectural innovations culminating in large-scale foundation models capable of language generation, visual reasoning, code synthesis, and scientific discovery. This systematic narrative review provides a comprehensive, pedagogically rigorous exposition of the principal architectural families in deep learning, organized as an intellectual genealogy in which each advance is motivated by the specific limitations of its predecessor. The review covers convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), gated recurrent units (GRUs), and the complete taxonomy of attention mechanisms—including self-attention, multi-head attention, cross-attention, masked attention, global and local attention, sparse attention, Flash Attention, linear attention, and rotary positional embeddings—before developing the full Transformer architecture and its landmark instantiations: BERT, GPT, GPT-3, T5, and Vision Transformers. The review then extends to generative AI architectures, including denoising diffusion probabilistic models and latent diffusion models, and to alignment techniques including reinforcement learning from human feedback (RLHF), Constitutional AI, and direct preference optimization (DPO). Theoretical properties of the Transformer—universal approximation, Turing completeness under idealized conditions, and counter-machine equivalence under practical constraints—are treated with explicit delineation of ideal versus practical expressivity. Scaling laws are presented with appropriate epistemic qualification, and the contested nature of emergent abilities is addressed by incorporating the Schaeffer et al. (2023) metric-artifact critique alongside the original Wei et al. (2022b) observations. In-context learning is interpreted as implicit Bayesian inference, and mechanistic interpretability findings—including induction heads and indirect object identification circuits—

are reviewed. Limitations including hallucination, long-context faithfulness failures, quadratic complexity, and inference-cost constraints are analyzed, followed by a discussion of open-weight versus closed-weight model accessibility. The review is grounded in more than 65 peer-reviewed and preprint references spanning 1943 to 2024 and is structured to serve as both an introductory guide for novice learners and a foundational reference for practitioners and researchers entering the field.

Keywords: *neural networks, convolutional neural networks, recurrent neural networks, LSTM, attention mechanisms, Transformer, self-attention, multi-head attention, diffusion models, large language models, scaling laws, emergent abilities, in-context learning, mechanistic interpretability, alignment, RLHF*

Review Methodology

This article constitutes a systematic narrative review. Literature was identified through searches of Google Scholar, the Semantic Scholar Corpus, and the arXiv preprint server, using the following primary search terms and their combinations: neural network, convolutional neural network, recurrent neural network, long short-term memory, attention mechanism, Transformer architecture, self-attention, large language model, foundation model, diffusion model, scaling laws, emergent abilities, in-context learning, mechanistic interpretability, and reinforcement learning from human feedback. Searches were conducted in January and February 2026 and were bounded by publications dated January 1, 1943, through February 1, 2026.

Inclusion criteria required that sources: (a) constitute original empirical research, theoretical analysis, or a well-recognized review published in a peer-reviewed venue or a widely-cited preprint; (b) are directly relevant to at least one of the architectural families, theoretical properties, or application domains enumerated in the title; and (c) are cited a minimum of 50 times in the Semantic Scholar corpus (with exceptions granted for works published after January 2023, for which citation counts are still accumulating). Exclusion criteria eliminated purely engineering tutorial materials, secondary survey articles that did not contribute novel synthesis, and works that duplicated results already represented by a higher-impact primary source. Sixty-eight sources meeting these criteria are cited in the final review. The review makes no claim to exhaustive coverage of the literature; rather, it selects canonical and landmark contributions that serve the pedagogical and synthetic purposes articulated in the abstract.

Preliminaries: Mathematical and Machine Learning Foundations

This section establishes the mathematical and conceptual vocabulary used throughout the review. Readers with a background in linear algebra and introductory probability may proceed directly to the Introduction; those without this background are encouraged to read this section carefully before continuing.

Vectors, Matrices, and Dot Products

A vector is an ordered list of real numbers. A vector x of dimension d is written $x \in \mathbb{R}^d$, and its i-th entry is denoted x_i. A matrix X of dimensions n × d is an array with n rows and d columns; entry X_{ij} denotes the element at row i and column j. In the context of this review, each row X_i represents the d-dimensional embedding of the i-th token in a sequence of n tokens.

The dot product of two vectors $u, v \in \mathbb{R}^d$ is the scalar $u \cdot v = \Sigma_i u_i v_i$. Geometrically, the dot product measures the degree of alignment between the two vectors: a large positive value indicates that u and v point in similar directions; a value near zero indicates near-orthogonality. In attention mechanisms, dot products are used to measure how relevant one token is to another.

Matrix multiplication AB, where $A \in \mathbb{R}^{nxk}$ and $B \in \mathbb{R}^{kxd}$, produces a matrix $C \in \mathbb{R}^{nxd}$ where $C_{ij} = \Sigma_k A_{ik} B_{kj}$. In neural networks, each weight matrix W defines a linear transformation: the product xW maps a vector x through a learned linear projection.

Activation Functions

Neural networks achieve expressive power by composing linear transformations with nonlinear activation functions. The sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ maps any real number to the open interval (0, 1) and is used in gating mechanisms where the output represents a proportion—how much to let through. The hyperbolic tangent $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ maps to (−1, 1) and is used to produce bounded, signed activations. The rectified linear unit ReLU(x) =

max(0, x) is computationally simple, avoids the vanishing gradient problem for positive inputs, and has become the default nonlinearity in feedforward sublayers.

Softmax and Probability Distributions

The softmax function converts a vector of arbitrary real values $z \in \mathbb{R}^k$ into a valid probability distribution:

$$softmax(z_i) = exp(z_i) / \Sigma_j\ exp(z_j), \quad for\ i = 1, ..., k$$

The output sums to 1 and each entry is positive. In attention mechanisms, softmax is applied row-wise to a matrix of attention logits: each row of the attention matrix corresponds to one query position, and the softmax for that row produces the probability distribution over key positions for that specific query. This row-wise application is essential—it ensures that each query independently sums its attention weights to 1, rather than conflating attention across queries.

Parameters, Training, and Generalization

A neural network is a parameterized function $f\theta(x)$ where θ denotes all learnable weights. Training minimizes a loss function $L(\theta)$ that measures the discrepancy between the network's predictions and the correct outputs on a training dataset. The dominant optimization algorithm in deep learning is Adam (Kingma & Ba, 2015), which maintains adaptive per-parameter learning rates based on first and second moment estimates of the gradient. Training typically proceeds with a warmup phase during which the learning rate is increased linearly from near zero to a peak value, followed by a decay phase. In Transformer training, this warmup is essential for stability: without it, the large random gradients in early training can destabilize the layer normalization (Vaswani et al., 2017).

Generalization refers to a model's ability to perform well on data not seen during training. Overfitting occurs when a model learns to reproduce the training data rather than the underlying

distribution. Dropout (Srivastava et al., 2014) is the most widely used regularization technique for deep networks: during training, random units are set to zero with probability p (typically 0.1 to 0.5), forcing the network to learn redundant representations that do not rely on any single unit.

Fine-Tuning, Zero-Shot, and Few-Shot Learning

A pretrained model is one that has been trained on a large corpus without supervision specific to any single task. Fine-tuning adapts a pretrained model to a target task by continuing training on a smaller labeled dataset, updating some or all of the model's parameters. Zero-shot learning refers to the ability of a model to perform a task from a natural language description of that task alone, without any task-specific examples or gradient updates. Few-shot learning provides a small number of labeled examples (the 'shots') in the prompt and asks the model to generalize from them, also without gradient updates. The distinction between zero-shot and few-shot performance is empirically important: large language models often perform near chance on zero-shot evaluations but dramatically improve with even two or three examples in the context.

Perplexity and Loss

Language model quality is often measured by perplexity, defined as the exponentiated average negative log-likelihood of the model assigning the correct next token: $\mathrm{PPL} = \exp(-(1/T)\,\Sigma_t \log P(x_t \mid x_1, \ldots, x_{t-1}))$. A lower perplexity indicates a more confident and accurate model. When scaling law papers report 'loss,' they refer to this average negative log-likelihood before exponentiation.

Introduction: The Genealogy of Neural Architectures

The history of artificial neural networks is a history of successive confrontations with the limits of preceding ideas. McCulloch and Pitts (1943) formalized the first mathematical model of a neuron. Rosenblatt (1958) demonstrated that a perceptron could learn linear decision boundaries from data. Rumelhart et al. (1986) showed that multilayer networks could be trained by backpropagating error gradients, enabling nonlinear function approximation. Yet despite these advances, two categories of structured data resisted effective neural treatment for decades: spatial data such as images, and sequential data such as text, speech, and time series.

Images carry rich local structures—edges, textures, and spatially organized parts—that demand architectures with built-in spatial inductive biases. Sequential data carries temporal dependencies that may span arbitrary distances: the grammatical subject of a sentence may determine the interpretation of a word fifty tokens later, and a melody's tonic may structure its resolution hundreds of notes ahead. Standard feedforward networks, which process each input dimension independently, are ill-equipped for both challenges.

Convolutional neural networks (CNNs), developed by LeCun et al. (1989, 1998) and spectacularly validated by Krizhevsky et al. (2012), resolved the image problem through weight sharing and local receptive fields. Recurrent neural networks (RNNs), formalized by Elman (1990) and Jordan (1986), resolved the sequence problem by maintaining a hidden state across time steps. Long short-term memory networks (LSTMs; Hochreiter & Schmidhuber, 1997) resolved the RNN's vanishing gradient problem through gated memory cells. Yet LSTMs remained fundamentally sequential, preventing parallelization on GPU hardware.

The decisive paradigm shift arrived in 2017. Vaswani et al. published 'Attention Is All You Need,' introducing the Transformer—a fully parallel, attention-based architecture that resolved the

sequential bottleneck while simultaneously capturing long-range dependencies. Building on an earlier attention mechanism for neural machine translation (Bahdanau et al., 2015), the Transformer replaced recurrence with self-attention: a mechanism allowing every element of a sequence to attend to every other element in a single parallel computation.

Yet the Transformer itself had intellectual precursors that a historically accurate account must acknowledge. Graves et al. (2014) introduced the Neural Turing Machine (NTM), which implemented content-addressable differentiable memory with attention-like read and write heads—a direct conceptual antecedent of the query-key-value formalism. Weston et al. (2015) proposed Memory Networks, which stored information in external memory arrays and retrieved it via soft attention. Mnih et al. (2014) drew on cognitive neuroscience models of spatial visual attention to develop recurrent attention models for image understanding. The Transformer can be understood as the architectural synthesis that made these ideas fully differentiable, fully parallel, and scalable to billions of parameters.

The decade following the Transformer's introduction produced a cascade of landmark systems. BERT (Devlin et al., 2019) established bidirectional pretraining for language understanding. GPT-3 (Brown et al., 2020) demonstrated that scaling a decoder-only language model to 175 billion parameters produces striking few-shot generalization. T5 (Raffel et al., 2020) unified NLP under a text-to-text framework. Vision Transformers (Dosovitskiy et al., 2021) demonstrated that convolutional inductive biases are unnecessary given sufficient data. Denoising diffusion probabilistic models (Ho et al., 2020) and latent diffusion models (Rombach et al., 2022) applied Transformer components to generative image synthesis, producing systems whose outputs are deployed in millions of daily interactions. Alignment techniques including RLHF (Ouyang et

al., 2022), Constitutional AI (Bai et al., 2022), and direct preference optimization (Rafailov et al., 2023) addressed the challenge of making these systems safe and helpful.

This review traces the complete arc of these developments. Each architectural family is introduced through the specific limitation it was designed to overcome, with mathematical foundations developed accessibly and theoretical results presented with explicit delineation of ideal from practical scope. The review is intended to serve as a pedagogical foundation for novice learners and a synthesis for researchers seeking a unified account of the field from its origins through the current frontier.

Convolutional Neural Networks

The Problem of Spatial Data

Consider a grayscale image of 256 × 256 pixels. Flattened into a vector, this image has 65,536 dimensions. A single fully connected layer mapping this input to 1,000 hidden units requires 65,536,000 weight parameters. The parameter count is not merely expensive; it is architecturally wrong. A feedforward network treats each pixel as an independent dimension, ignoring the fact that adjacent pixels share structural relationships. An edge is defined by local contrast between neighboring pixels, not by their absolute positions in a flattened vector. Crucially, a cat in the upper-left corner of an image and a cat in the lower-right corner are the same object and should be recognized identically—a property called translation invariance that feedforward networks cannot learn without enormous data augmentation.

Convolutional neural networks (CNNs), pioneered by LeCun et al. (1989, 1998) and inspired in part by Hubel and Wiesel's (1962) discoveries about orientation-selective cells in the mammalian visual cortex, address both problems through two mechanisms: local connectivity and weight sharing.

The Convolution Operation

A convolution slides a small learnable matrix called a filter or kernel across the input, computing dot products at each spatial position. For a two-dimensional image I and filter F of dimensions k × k, the output feature map S is:

$$S(i, j) = \Sigma_m \Sigma_n \; I(i + m, j + n) \cdot F(m, n)$$

Each position in S captures how strongly the filter pattern is present in the local image patch centered at (i, j). Different filters detect different features: a horizontal-edge filter produces high activation at horizontal boundaries; a Gabor-like filter responds to oriented textures.

Critically, the same filter weights are applied at every spatial position—weight sharing—so the number of parameters is determined by the filter size, not the input resolution. A layer with 64 filters of size 3 × 3 applied to a 3-channel (RGB) input requires only 64 × 3 × 3 × 3 + 64 = 1,792 parameters regardless of image resolution.

Pooling, Hierarchical Representation, and Normalization

A pooling layer reduces spatial resolution while retaining salient activations. Max pooling selects the maximum value within each non-overlapping window; average pooling computes the mean. Pooling provides spatial invariance within small neighborhoods and reduces the computational burden for subsequent layers.

The power of CNNs lies in hierarchical composition. Early layers detect low-level features (edges, corners, gradients). Middle layers assemble these into part representations (eyes, wheels, textures). Deep layers represent whole objects or semantic concepts. This hierarchy of abstraction, noted by LeCun et al. (2015) as biologically inspired, produces increasingly abstract representations with each layer.

Modern CNNs incorporate batch normalization (Ioffe & Szegedy, 2015) after convolutional layers, normalizing each feature map's activations across the batch to stabilize training and accelerate convergence—a technique conceptually related to layer normalization in Transformers.

The AlexNet Revolution and Subsequent Architectures

Despite theoretical development in the 1980s and 1990s, CNNs did not dominate image recognition until Krizhevsky et al. (2012) trained AlexNet on ImageNet using GPU hardware. AlexNet achieved a top-5 error rate of 15.3% on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), compared with 26.2% for the next best method—a margin of more than 10

percentage points that the community recognized as decisive. AlexNet introduced ReLU activations, dropout regularization (Srivastava et al., 2014), and GPU-accelerated training as standard practice.

Subsequent architectures drove the error rate to near-human levels. VGGNet (Simonyan & Zisserman, 2014) demonstrated that very deep networks with small (3 × 3) filters consistently outperform shallower networks with larger filters. GoogLeNet (Szegedy et al., 2015) introduced Inception modules that compute convolutions at multiple scales in parallel. ResNet (He et al., 2016) introduced residual connections—shortcut paths from input to output of each block—that provided direct gradient pathways enabling reliable training of networks with 152 layers. The ResNet-152 achieves a top-5 error rate of 3.57% on ImageNet, below the estimated human performance of approximately 5%.

Limitations Motivating Sequential Architectures

CNNs excel at spatial feature extraction but impose constraints that motivate subsequent architectural exploration. The receptive field of a convolutional layer grows linearly with depth; capturing global context requires very deep stacking. Pooling is lossy, discarding precise spatial relationships. Most significantly, CNNs carry no inductive bias for temporal structure: while one-dimensional convolutions can be applied to sequences, they lack any mechanism for modeling dependencies that span variable distances in time. These limitations—particularly for language and speech—motivated the development of recurrent architectures.

Recurrent Neural Networks

Motivation and Hidden State

Human language is irreducibly sequential. The meaning of each word depends on what preceded it, and the correct interpretation of a sentence often cannot be determined until its end. Speech unfolds as a waveform through time; financial time series encode structural dependencies across months or years. A feedforward network presented with each token of a sentence in isolation—with no memory of prior tokens—cannot model these dependencies. Recurrent neural networks (RNNs) address this by maintaining a hidden state $h_t \in \mathbb{R}^H$—a vector that accumulates a compressed representation of all prior inputs—and updating it at each time step.

Formal Definition

Given an input sequence $x_1, x_2, ..., x_t$, the RNN computes:

$$h_t = tanh(W\_hh \cdot h_{t-1} + W\_xh \cdot x_t + b\_h)$$

$$y_t = W\_hy \cdot h_t + b\_y$$

where $W\_hh \in \mathbb{R}^{HxH}$ is the recurrent weight matrix, $W\_xh \in \mathbb{R}^{Hxd}$ projects the input, b_h is a bias vector, and y_t is the output at step t. The same weight matrices are applied at every time step, so the total parameter count is fixed regardless of sequence length. Outputs may be produced at every step (sequence labeling) or only at the final step (classification).

Elman (1990) demonstrated that RNNs trained by backpropagation could discover grammatical structure in language data. Mikolov et al. (2010) applied RNNs to language modeling, achieving perplexity improvements over n-gram baselines on standard benchmarks.

Backpropagation Through Time and the Vanishing Gradient Problem

RNNs are trained using backpropagation through time (BPTT), in which the network is unrolled across all time steps and gradients are propagated backward from the final loss. To

compute the gradient of the loss L with respect to the hidden state at time t_0, the chain rule requires multiplying Jacobians of the hidden state transition across all subsequent steps:

$$\partial L/\partial h_{t_0} = (\partial L/\partial h_T) \cdot \Pi_{t=t_0+1}^{T} (\partial h_t/\partial h_{t-1})$$

Each factor $\partial h_t/\partial h_{t-1} = \text{diag}(1 - h_t^2) \cdot W_{hh}$ is a matrix whose spectral norm is governed by the eigenvalues of W_hh. Bengio et al. (1994) formally demonstrated that when the dominant eigenvalue of W_hh is less than 1, this product of matrices decays exponentially in $T - t_0$, causing gradient signals to vanish; when it is greater than 1, gradients grow exponentially, causing numerical explosion. Under standard training conditions—without specialized initialization—RNNs reliably learn dependencies spanning approximately 10 to 20 time steps, beyond which the gradient signal is too weak to drive learning (Bengio et al., 1994). Gradient clipping (Pascanu et al., 2013) mitigates the exploding case but cannot recover vanishing gradients.

Long Short-Term Memory Networks and Gated Recurrent Units

The LSTM Architecture

The long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) resolves the vanishing gradient problem by introducing a cell state $C_t \in \mathbb{R}^H$—a vector that flows through time with minimal transformation, controlled by three sigmoid-activated gates. Each gate outputs values in (0, 1), functioning as a learned scalar regulator: a value near 1 allows information to pass; near 0, it blocks it.

The forget gate f_t determines what proportion of the previous cell state to retain:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The input gate i_t and candidate cell state $\tilde{C}_t$ determine what new information to write:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The cell state is updated additively:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

The output gate o_t produces the hidden state by filtering the squashed cell state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

In these equations, σ denotes the sigmoid function, $\odot$ denotes element-wise multiplication, and $[h_{t-1}, x_t]$ denotes vector concatenation of the previous hidden state and current input. The biases (b_f, b_i, b_C, b_o) are omitted from some compact presentations for brevity but are present in all standard implementations.

The Gradient Highway: Why the Cell State Resolves Vanishing Gradients

The additive cell state update $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$ is the LSTM's critical innovation. Backpropagating through this additive path gives:

$$\partial C_t / \partial C_{t-1} = diag(f_t)$$

This is a diagonal matrix with entries equal to the forget gate values $f_{t,i} \in (0, 1)$. Crucially, unlike the RNN's full non-diagonal Jacobian W_hh—which compresses and rotates gradient vectors in ways that cause their norm to decay—the LSTM's cell state gradient passes through only a diagonal scaling. When f_t is kept near 1 (forget gate open), the gradient of the loss with respect to C_{t0} is multiplied only by diagonal factors close to 1 across all time steps. This is what Hochreiter and Schmidhuber (1997) called the Constant Error Carousel: gradients can flow arbitrarily far back through time with essentially no decay, enabling reliable learning of dependencies spanning hundreds or thousands of time steps. The hidden state h_t still passes through the tanh and output gate, but the cell state path provides a clean gradient highway that supports effective backpropagation across the full sequence length. The output gate correctly first squashes the cell state through tanh and then filters it element-wise by o_t—not the reverse.

Empirical Validation

The LSTM's long-range learning capability was validated on diverse tasks. Graves et al. (2013) applied stacked LSTMs to speech recognition, achieving then-state-of-the-art results on the TIMIT phoneme recognition benchmark. Sutskever et al. (2014) used stacked LSTMs in the sequence-to-sequence encoder-decoder framework to achieve state-of-the-art machine translation on WMT English-French and English-German evaluation sets. These results established the LSTM as the dominant sequence modeling architecture from approximately 2014 to 2017.

Gated Recurrent Units

The gated recurrent unit (GRU; Cho et al., 2014) simplifies the LSTM by merging the cell state and hidden state into a single vector and combining the forget and input gates into an update gate z_t:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The reset gate r_t controls how much of the previous hidden state contributes to the candidate activation; when $r_t \approx 0$, the GRU effectively starts fresh, ignoring prior history. The GRU achieves comparable performance to the LSTM on many standard benchmarks with fewer parameters, making it attractive for memory-constrained applications. Note that the bias terms are omitted from the GRU equations above for clarity but are standard in all implementations.

Persistent Limitations

Despite their considerable advances over vanilla RNNs, LSTMs and GRUs retain two fundamental limitations. First, processing a sequence of length T requires T sequential operations; because each state h_t depends on h_{t-1}, no time step can begin until the previous one completes. This sequential constraint prevents parallelization on GPU hardware, which is optimized for massively parallel operations. Training LSTMs on long sequences is therefore much slower than training CNNs of comparable parameter count. Second, even the LSTM's cell state is a fixed-size vector that must compress all information relevant to generating any future output. For very long sequences—multi-document contexts, entire codebases, long-form narratives—this fixed-capacity compression increasingly loses important detail. Both limitations—sequential computation and fixed-size memory—would be directly targeted by the attention mechanism.

Tokenization and Vocabulary

All architectures reviewed in this paper operate on sequences of discrete units called tokens. Understanding what a token is, and how raw text is converted into tokens, is a prerequisite for connecting the mathematical formalisms to actual data.

The Tokenization Problem

Raw text is a sequence of characters from a finite alphabet. The simplest tokenization is character-level: each character is its own token. Character-level models can represent any text without out-of-vocabulary issues but must learn long-range dependencies at a very fine granularity, requiring longer sequences for equivalent context. Word-level tokenization uses whitespace-delimited words as tokens, producing shorter sequences but failing on rare or unseen words and on morphologically rich languages.

The dominant approach in modern large language models is subword tokenization. The most widely used algorithm is byte pair encoding (BPE; Sennrich et al., 2016), which begins with a character-level vocabulary and iteratively merges the most frequent pair of adjacent tokens into a new token. BPE produces a vocabulary of fixed size (typically 30,000 to 100,000 tokens) that covers common words as single tokens, rare words as sequences of subword units, and arbitrary byte sequences as a fallback. A related approach, unigram language model tokenization (Kudo & Richardson, 2018), selects the vocabulary that maximizes the likelihood of the training corpus under a probabilistic model.

Tokenization in Practice

The GPT family uses BPE with a vocabulary of approximately 50,000 tokens. The LLaMA models use SentencePiece (Kudo & Richardson, 2018) with a vocabulary of 32,000 tokens. BERT uses WordPiece (Schuster & Nakamura, 2012), a variant of BPE with likelihood-based merging.

Multimodal models tokenize images by dividing them into fixed-size patches (typically 14 × 14 or 16 × 16 pixels) and projecting each patch into the token embedding space, as in the Vision Transformer (Dosovitskiy et al., 2021).

The choice of tokenization has practical consequences beyond vocabulary size. Tokenization artifacts—such as different tokenizations of semantically equivalent strings—can affect model behavior in subtle ways. Numbers and code identifiers are particularly susceptible, because a number like 12345 may be split as [123, 45] or [1, 23, 45] depending on frequency statistics, leading to inconsistent mathematical reasoning. Understanding these artifacts is important for anyone building on or evaluating the models discussed in subsequent sections.

The Origins of Attention: From Cognitive Science to Sequence Modeling

Precursors in Cognitive Science and Neuroscience

The term attention in computational models draws directly from cognitive psychology and neuroscience. Selective attention—the organism's ability to allocate processing resources preferentially to relevant stimuli—is one of the most studied phenomena in cognitive science (Treisman & Gelade, 1980). Feature-based attention and spatial attention are distinct neural mechanisms in the primate visual system. Mnih et al. (2014) explicitly invoked these findings in developing the recurrent attention model for image understanding, demonstrating that a model that learned to attend to informative image regions in sequence could perform visual object recognition with fewer computations than a model that processed the entire image uniformly. This work represents one of the clearest bridges between cognitive science and modern computational attention.

The Neural Turing Machine (NTM; Graves et al., 2014), published a year before Bahdanau et al. (2015), introduced a fully differentiable content-addressable external memory. The NTM's read and write heads produced a softmax distribution over memory locations based on a content similarity score between a query vector and each memory slot—a mechanism structurally identical to the query-key-value formalism that would define all subsequent attention research. Memory Networks (Weston et al., 2015), published concurrently with Bahdanau, stored facts in an external memory array and retrieved them by computing a soft attention distribution over the array in response to a query. These works established the conceptual vocabulary of attention before the Transformer; attributing the query-key-value formalism to Bahdanau alone would misrepresent the collaborative nature of scientific development.

The Sequence-to-Sequence Bottleneck

By 2014, the dominant neural machine translation framework was sequence-to-sequence modeling (Sutskever et al., 2014), in which an encoder LSTM processes the source sentence and compresses it into a single fixed-length context vector, which is then passed to a decoder LSTM that generates the target sentence token by token. Sutskever et al. demonstrated strong results on short sentences but acknowledged that performance degraded sharply with increasing source length. A single fixed-length vector—typically 256 to 1,024 dimensions—must encode all the semantic, syntactic, and positional information of an arbitrarily long source sentence. For paragraphs or documents, this constitutes a catastrophic information bottleneck that the fixed capacity of the context vector cannot resolve.

Bahdanau Attention

Bahdanau et al. (2015) resolved this bottleneck by allowing the decoder to attend selectively to the encoder's full sequence of hidden states at each decoding step, rather than being limited to a single compressed vector. The encoder (a bidirectional RNN) produces annotations h_1, h_2, ..., h_n, one per source position. At decoder time step t, an alignment score is computed between the decoder's previous hidden state s_{t-1} and each encoder annotation h_j via a small feedforward score function a:

$$e_{tj} = a(s_{t-1}, h_j)$$

These scores are normalized by softmax into attention weights:

$$\alpha_{tj} = \exp(e_{tj}) \;/\; \Sigma_k \exp(e_{tk})$$

The context vector for step t is the weighted sum of encoder annotations:

$$c_t = \Sigma_j \alpha_{tj} \cdot h_j$$

Visualizing α_{ij} as a heatmap over source positions reveals interpretable alignment patterns: when translating 'chien' (dog) into French, α_{ij} is high at the English position corresponding to 'dog'

and low elsewhere. Bahdanau et al. (2015) demonstrated that this mechanism substantially improved translation quality on longer sentences compared with the fixed-context baseline.

Luong Attention and the QKV Abstraction

Luong et al. (2015) simplified the additive score function of Bahdanau with a multiplicative dot-product form, demonstrated superior computational efficiency, and proposed computing the context vector after—rather than before—the decoder state update. These simplifications contributed to a modular reformulation of attention as a function of three abstract quantities: a query (what is the decoder seeking?), a set of keys (what does each source position contain?), and a set of values (what does each source position contribute?). This query-key-value abstraction, now standard, makes the mechanism's structure transparent and composable.

Self-Attention

Definition and Formalization

Self-attention, or intra-attention, applies the attention mechanism within a single sequence: every element attends to every other element in the same sequence, including itself. Unlike Bahdanau's encoder-decoder attention, self-attention generates queries, keys, and values from the same input.

Given an input sequence represented as a matrix $X \in \mathbb{R}^{nxd}$, where n is the number of tokens and d is the embedding dimension, three learned linear projections produce:

$$Q = X \cdot WQ, \qquad K = X \cdot WK, \qquad V = X \cdot WV$$

where $WQ, WK, WV \in \mathbb{R}^{dxdk}$ are weight matrices projecting the input into query, key, and value spaces respectively. The attention output is:

$$Attention(Q, K, V) = softmax(Q \cdot KT / \sqrt{d_k}) \cdot V$$

The softmax is applied row-wise to the n × n matrix of attention logits: each row i produces an independent probability distribution α_i over all n positions, representing how much position i attends to each other position. This row-wise application is essential—each query position must independently sum its attention weights to 1, not share a normalization with other queries. Division by $\sqrt{d_k}$ prevents the dot products from growing so large that the softmax saturates into near-zero gradients; in high-dimensional spaces, random dot products have variance proportional to d_k, and dividing by $\sqrt{d_k}$ restores unit variance (Vaswani et al., 2017).

The output for position i is the weighted sum of value vectors:

$$o_i = \Sigma_j \alpha_{ij} \cdot v_j$$

This is a contextualized representation of position i that integrates information from all other positions—including itself—weighted by relevance.

A Worked Example

Consider the sentence: 'The animal did not cross the road because it was too tired.' When computing the self-attention representation of the pronoun 'it,' the model should assign high α_{ij} to 'animal' (the correct referent) and lower values to 'road' (a competing noun) and other positions. Visualizations confirm that trained Transformer models exhibit exactly this pattern (Clark et al., 2019), demonstrating that self-attention can capture the coreference dependency between 'it' and 'animal' regardless of the distance between them in the sequence—a dependency that vanilla RNNs often fail to model reliably.

Computational Complexity

Self-attention requires computing all $n \times n$ pairwise attention scores, giving $O(n^2d)$ time complexity and $O(n^2)$ memory complexity. For a sequence of length 512 (standard for BERT), this requires 262,144 pairwise computations. For sequences of length 32,768 (long-context models), the quadratic cost becomes prohibitive without algorithmic innovations, motivating the efficient attention variants developed in Sections 12–14.

Permutation Equivariance and Positional Encoding

Self-attention is permutation equivariant: permuting the input rows of X permutes the output rows of the attention result correspondingly, but does not change the computation itself. This means self-attention has no intrinsic notion of order or distance—the word at position 1 and the word at position 50 are treated identically in the absence of positional information. Positional encodings, discussed in depth in Section 15, are injected into the input embeddings to restore positional awareness.

Multi-Head Attention

Motivation

A single attention head computes one weighted combination of value vectors, producing a single perspective on the relational structure of the input. This single head must simultaneously capture syntactic dependencies, semantic associations, coreference relations, and positional proximity—competing demands that a single projection may inadequately balance. Multi-head attention (Vaswani et al., 2017) runs h independent attention operations in parallel, each in a distinct lower-dimensional projection of the input, then concatenates and projects their outputs.

Formal Definition

With h heads, each head i operates on projections of dimension $d_k = d_{out}/h$:

$$head_i = Attention(Q \cdot W_iQ, K \cdot W_iK, V \cdot W_iV)$$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \cdot WO$$

where $W_iQ, W_iK, W_iV \in \mathbb{R}^{dmodelxdk}$ are per-head projection matrices and $WO \in \mathbb{R}^{hdkxdmodel}$ is the output projection. The four parameter matrices WQ, WK, WV, and WO together contain approximately $4 \times d_{mo}{}^{d}{}_{el}{}^{2}$ parameters—equivalent to a single-head attention layer with an output projection. The parameter count is identical; the gain is representational diversity across h specialized subspaces.

What Individual Heads Learn

Clark et al. (2019) analyzed BERT's 144 attention heads (12 layers × 12 heads) and found that different heads specialize in distinct linguistic patterns: some attend primarily to direct objects, others to syntactic subjects, others to tokens within the same noun phrase, others to the immediately preceding token. Voita et al. (2019) demonstrated that roughly 70–80% of attention heads can be pruned with negligible performance loss, confirming that the remaining heads

perform specific, non-redundant computations. These findings establish that multi-head attention is not a homogeneous averaging operation but a compositional system of specialized relational operators.

Connection to Ensemble Learning

Multi-head attention is conceptually analogous to ensemble methods in classical machine learning: multiple diverse learners are combined to reduce variance and improve robustness. Each head specializes on a different projection of the relational space, and their concatenated outputs provide a richer, multi-perspective token representation than any single head could produce alone.

Cross-Attention, Masked Attention, and Global-Local Attention

Cross-Attention

Cross-attention is the variant in which queries originate from one sequence (the target) and keys and values from a different sequence (the source). In the encoder-decoder Transformer, the decoder's current representation serves as queries while the encoder's output serves as keys and values:

$$Q = h_{dec} \cdot W_Q, \qquad K = h_{enc} \cdot W_K, \qquad V = h_{enc} \cdot W_V$$

At each decoder position, the query asks: given my current state, which parts of the source sequence are most relevant? The encoder keys and values answer by providing the content of each source position. This mechanism generalizes directly to multimodal settings: in CLIP (Radford et al., 2021), cross-attention aligns textual queries with visual patch representations; in latent diffusion models (Rombach et al., 2022), cross-attention conditions the image denoising process on text embeddings, allowing the text prompt to steer the generated image at each denoising step. The cross-attention formulation in Section 7 (Bahdanau attention) was thus not only a solution to the machine translation bottleneck but the foundational primitive for conditioning in all modern generative models.

Masked Attention and Autoregressive Generation

Full self-attention allows each position to attend to all positions including future ones. For autoregressive generation—where the model must predict each token given only preceding tokens—attending to future tokens constitutes information leakage that invalidates the probabilistic factorization of the sequence. Masked (causal) attention enforces a strictly lower-triangular attention pattern by adding $-\infty$ to future logits before softmax:

$$\text{Masked-Attn}(Q, K, V) = \text{softmax}((Q \cdot K^T + M) / \sqrt{d_k}) \cdot V$$

where $M_{ij} = 0$ if $j \leq i$ and $M_{ij} = -\infty$ if $j > i$. After softmax, the $-\infty$ entries map to zero probability, effectively blocking future attention. This mechanism defines two major Transformer model classes: decoder-only models (GPT, LLaMA) use causal masking throughout and are capable of autoregressive generation; encoder-only models (BERT) use no causal masking and are capable of bidirectional contextualization but not direct generation. BERT instead uses masked language modeling (MLM) as its pretraining objective, masking 15% of input tokens and training the model to predict them from fully bidirectional context.

Global and Local Attention

Standard self-attention—every token attending to every other token—provides the richest possible relational modeling but at $O(n^2)$ cost. Local (windowed) attention restricts each token to a window of w neighbors, reducing cost to $O(nw)$. Information from distant positions must propagate through stacked layers, each extending the effective receptive field by w positions—analogously to hierarchical CNNs.

Beltagy et al. (2020) proposed the Longformer, combining local windowed attention with a small number of global tokens (the [CLS] classification token and task-specific tokens) that attend to all positions. The global tokens serve as aggregation nodes through which long-range information flows without requiring all tokens to attend globally. This hybrid achieves linear complexity while retaining global representational capacity, enabling effective processing of documents up to 4,096 tokens on tasks including scientific question answering and legal text analysis. Zaheer et al. (2020) proved theoretically that their Big Bird architecture—combining local windowed, global, and random attention—preserves the expressivity of full attention while achieving linear complexity.

Sparse Attention

Structured Sparsity

Full self-attention is statistically as well as computationally inefficient: empirical analysis of trained models reveals that attention weight matrices are highly sparse in practice, with most attention mass concentrated in a small fraction of positions (Clark et al., 2019; Voita et al., 2019). Sparse attention methods exploit this by enforcing structured sparsity—restricting attention to a predetermined or dynamically chosen subset of positions—rather than computing the full dense attention matrix.

Sparse Transformers

Child et al. (2019) introduced the Sparse Transformer, which alternates between two complementary attention patterns across successive layers: a local pattern attending to the w nearest positions, and a strided pattern attending at intervals of stride s. Together, any two tokens can interact within two layers: the strided connections bridge long distances while the local connections provide fine-grained context. Sparse Transformers enabled training on sequences of 12,000 to 64,000 tokens—previously intractable—on tasks including long-range music modeling and document generation.

Routing-Based Approaches: Reformer and Performer

Fixed sparsity patterns are input-independent, potentially missing the most informative connections for a given input. Kitaev et al. (2020) introduced the Reformer, which uses locality-sensitive hashing (LSH) to dynamically route each query to the most similar keys, attending only where attention weights would naturally be highest. This content-based routing adapts the sparsity pattern to the input, recovering the flexibility of full attention at reduced cost. Choromanski et al. (2021) proposed the Performer, which uses random feature approximations of the softmax kernel

to achieve linear-complexity attention without any hard sparsity—the approximation is smooth and differentiable, preserving gradient flow throughout training.

Flash Attention

The Memory Bandwidth Bottleneck

Standard attention implementations materialize the full n × n attention matrix in GPU high-bandwidth memory (HBM). For a sequence of length 4,096 in 16-bit precision, this matrix requires approximately 32 MB per attention layer—a severe constraint at long sequence lengths. The deeper problem is not total memory capacity but memory bandwidth: the rate at which data can be transferred between HBM and the GPU's on-chip SRAM (shared memory and registers) is the binding computational constraint for attention, not arithmetic throughput. Modern GPU compute capabilities (measured in TFLOPS) have grown faster than memory bandwidth (measured in TB/s), creating a regime in which attention is memory-bandwidth-bound, not compute-bound.

The Tiling Algorithm and Online Softmax

Dao et al. (2022) introduced FlashAttention, which computes exact attention without materializing the full attention matrix in HBM. The key algorithmic insight is online softmax (Milakov & Gimelshein, 2018): it is possible to compute the softmax of a vector incrementally, processing one block at a time, by maintaining two running statistics—a running maximum m and a running normalization sum ℓ. As each new block of logits is processed, m and ℓ are updated and the contribution to the output sum is rescaled accordingly:

$$m_{new} = max(m_{old}, max(new\ logit\ block))$$

$$\ell_{new} = e^{(m_{old} - m_{new})} \cdot \ell_{old} + \Sigma\ e^{(logits - m_{new})}$$

This allows the exact softmax output to be computed without ever storing all logits simultaneously. FlashAttention tiles Q, K, and V into blocks that fit in SRAM, iterates over K and V blocks in an outer loop and Q blocks in an inner loop, applying the online softmax update at

each step and accumulating the output directly. Only the final output O is written to HBM—not the intermediate attention matrix. This eliminates the dominant HBM read/write cost.

Impact

FlashAttention reduced attention memory usage by 5 to 20 times and achieved wall-clock training speedups of 2 to 4 times on long sequences (≥ 2,048 tokens) compared with standard PyTorch attention implementations (Dao et al., 2022). On shorter sequences, the tiling overhead can marginally increase latency; the speedup is most dramatic at length 4,096 and beyond. FlashAttention-2 (Dao, 2023) improved parallelism across GPU thread blocks and achieved an additional two-times speedup. FlashAttention has been adopted by virtually all major LLM training frameworks and enabled practical training of models with context windows of tens of thousands of tokens.

Linear Attention and State-Space Models

Kernel Decomposition of Softmax Attention

The softmax attention computation $\text{softmax}(QK^T/\sqrt{d_k})V$ couples all query and key positions through a joint normalization, preventing the computation from being rearranged into a form that is sublinear in n. If the softmax could be approximated as a product of independent feature maps—such that $\text{softmax}(q \cdot k) \approx \varphi(q)^T \varphi(k)$ for some mapping φ—then the attention output could be computed as:

$$\text{Attention}(Q, K, V) \approx \varphi(Q) \cdot (\varphi(K)^T \cdot V)$$

By associativity of matrix multiplication, the matrix $\varphi(K)^T \cdot V \in \mathbb{R}^d \_ \varphi^{xd}$ can be computed once for all positions and then each query $\varphi(q_i)$ is multiplied against this shared summary. This reduces complexity from $O(n^2d)$ to $O(nd\varphi d)$, linear in n for fixed feature dimension $d\varphi$.

Katharopoulos Linear Attention and Autoregressive Reformulation

Katharopoulos et al. (2020) formalized this with the feature map $\varphi(x) = \text{elu}(x) + 1$, where elu is the exponential linear unit. They proved that linear attention permits an exact reformulation as an RNN during autoregressive inference: at each time step, the model updates a fixed-size state matrix $S = \Sigma_j \varphi(k_j) v_j^T$ incrementally and computes the output as $\varphi(q_t) \cdot S$, requiring $O(1)$ computation per step rather than $O(n)$ as in standard attention. This inference-time advantage is substantial for deployment at scale: standard attention requires $O(n)$ computation per new token (attending to all prior tokens), while linear attention requires $O(1)$ computation and $O(d^2)$ constant-size memory, making it particularly attractive for real-time generation and edge deployment.

Trade-Offs and Mamba

Linear attention sacrifices approximation quality: the elu feature map produces smoother, less peaked distributions than softmax, losing the sharp focus that enables precise information

retrieval. On benchmarks requiring accurate recall of specific items from long contexts (Tay et al., 2021), linear attention consistently underperforms exact softmax attention. The gap narrows for tasks tolerant of approximate retrieval.

Mamba (Gu & Dao, 2023) represents a distinct but related line of development based on structured state space models (SSMs; Gu et al., 2022) rather than linear attention. It is important to distinguish these: Mamba is not a linear attention model. It is a selective SSM in which the state transition matrices are input-dependent—the selection mechanism adapts the model's memory to the current input, filtering which information to propagate through the state. This input-dependent selection is what gives Mamba its competitive performance relative to Transformers on sequence modeling tasks. While the two approaches share linear complexity and a recurrent inference formulation, their architectural mechanics are fundamentally different. Scaling experiments (Waleffe et al., 2024) suggest that hybrid architectures combining Mamba layers with attention layers outperform pure SSMs at scales beyond approximately 3 billion parameters.

Positional Encodings

The Permutation Equivariance Problem

Self-attention is permutation equivariant: it produces the same outputs regardless of the order in which tokens are presented. The sentences 'The cat ate the mouse' and 'The mouse ate the cat' produce identical bag-of-token representations, yet their meanings are opposite. Positional encodings restore positional awareness by injecting information about each token's position into the model.

Sinusoidal Encodings

Vaswani et al. (2017) proposed adding fixed sinusoidal functions of position to input embeddings. For position pos and dimension index i ranging from 0 to $d_{model}/2 - 1$:

```
PE(pos, 2i)   = sin(pos / 10000^(2i / d_model))
PE(pos, 2i+1) = cos(pos / 10000^(2i / d_model))
```

The sinusoidal functions satisfy the property that PE(pos + k) is a linear function of PE(pos) for any fixed offset k, allowing the model to represent relative positions as linear transformations. Sinusoidal encodings also allow limited extrapolation to sequence lengths beyond the training maximum, though performance typically degrades for substantially longer sequences.

Learned Absolute Positional Embeddings

An alternative is to learn a position embedding table $E \in \mathbb{R}^{n_{max} x d}$, one embedding per position up to a maximum length. This is simpler and matches or exceeds sinusoidal encoding performance on standard-length tasks (Devlin et al., 2019). However, learned absolute embeddings cannot generalize to positions beyond n_{max}.

Relative Positional Encodings and ALiBi

Relative encodings represent the distance between tokens rather than their absolute positions. Shaw et al. (2018) add learned relative biases to attention logits based on the offset between query and key positions. Press et al. (2022) introduced Attention with Linear Biases (ALiBi), which adds a fixed, non-learned linear bias proportional to the distance between positions: score(i, j) = $q_i \cdot k_j - m \cdot |i - j|$, where m is a head-specific slope. ALiBi requires no positional embedding parameters and, crucially, extrapolates to sequences longer than those seen during training without additional fine-tuning—an important practical advantage for deploying models on documents longer than the training context window.

Rotary Positional Embeddings and Long-Context Extrapolation

Su et al. (2021) proposed Rotary Positional Embedding (RoPE), which rotates query and key vectors in two-dimensional subspaces as a function of their positions. For query at position m and key at position n, RoPE multiplies each by a rotation matrix R_m and R_n respectively, yielding:

$$(R_m q)^T (R_n k) = q^T R_{n-m} k$$

The dot product depends only on the relative offset n − m, not absolute positions, making RoPE a relative positional encoding that is applied through rotation rather than addition. RoPE has become the dominant positional encoding in modern open-weight LLMs including LLaMA (Touvron et al., 2023) and Mistral, due to its mathematical elegance, compatibility with efficient attention kernels, and strong performance.

However, the base RoPE does not extrapolate well to context lengths beyond those seen during training: the rotation frequencies are calibrated to the training length, and positions beyond this length produce rotation angles outside the range seen during training, degrading performance. YaRN (Peng et al., 2023) addresses this by scaling the rotary frequencies using an NTK-aware interpolation scheme that spreads the frequency spectrum to accommodate longer contexts without

fine-tuning on long-context data. NTK-aware scaling (Bloc97, 2023) is a closely related approach based on neural tangent kernel theory that modifies the RoPE base frequency. Both techniques enable models trained on 4,096-token contexts to generalize to 32,768 or 131,072 token contexts with minimal degradation—a practical necessity for modern long-context deployments.

The Transformer Architecture

Overview

The Transformer (Vaswani et al., 2017) synthesizes multi-head self-attention, feedforward sublayers, residual connections, and positional encodings into a fully parallel architecture that resolves the sequential bottleneck of RNNs while matching or exceeding LSTM performance on sequence transduction tasks. By replacing recurrent computation with attention over the entire input simultaneously, the Transformer enables GPU parallelization during training and resolves the vanishing gradient problem: gradients can flow directly between any two positions through the attention mechanism without passing through a chain of recurrent multiplications.

Encoder Architecture

The Transformer encoder consists of N identical layers (N = 6 in the original paper, 12 or 24 in BERT-base and BERT-large). Each layer contains two sublayers, each wrapped by a residual connection and layer normalization. Layer normalization (Ba et al., 2016) is defined as:

$$LayerNorm(x) = \gamma \cdot (x - \mu) / \sqrt{(\sigma^2 + \varepsilon)} + \beta$$

where μ and σ^2 are the mean and variance of x computed across the feature dimension, ε is a small constant for numerical stability, and γ and β are learned scale and bias parameters. Layer normalization stabilizes activations and allows training with large learning rates. The pre-norm variant—applying normalization before each sublayer rather than after—improves stability for very deep Transformers and is preferred in most modern implementations.

The first sublayer is multi-head self-attention (Section 9). The second sublayer is a position-wise feedforward network (FFN) applied independently to each token representation:

$$FFN(x) = max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2$$

where the inner dimension is typically $d_{ff} = 4 \times d_{model}$ and $\max(0, \cdot)$ is the ReLU activation. The FFN introduces nonlinearity and expands the per-token representational capacity beyond what linear self-attention projections provide. Together, each encoder layer performs: sublayer output = LayerNorm(x + Sublayer(x)).

Decoder Architecture

The Transformer decoder consists of N identical layers, each with three sublayers: masked multi-head self-attention (preventing attention to future tokens), cross-attention to the encoder output, and a feedforward network. The masked self-attention enables autoregressive generation; the cross-attention provides access to source representations. For language modeling without a separate encoder (e.g., GPT-class models), the decoder-only variant omits cross-attention entirely, leaving only masked self-attention and feedforward sublayers.

Dropout and Regularization

Dropout (Srivastava et al., 2014) is applied after each sublayer output and to attention weights in the original Transformer. During training, each element is independently set to zero with probability p (typically 0.1) and the remaining elements are scaled by $1/(1 - p)$ to preserve expected values. Dropout prevents co-adaptation of units and improves generalization, particularly for models trained on limited data. In very large models trained on massive corpora, dropout rates are often reduced or eliminated because regularization from data diversity dominates.

Benchmark Performance

The original Transformer (Vaswani et al., 2017) achieved a BLEU score of 28.4 on WMT 2014 English-to-German translation, surpassing all prior results by more than 2 BLEU points, and 41.0 BLEU on English-to-French—the highest reported at the time. Training required 3.5 days on 8 P100 GPUs, compared with weeks for LSTM-based models of comparable performance.

BERT, GPT, T5, and the Era of Foundation Models

The Pretrain–Fine-Tune Paradigm

The Transformer's scalability made pretraining dramatically more effective than prior transfer learning approaches for NLP. Peters et al. (2018) demonstrated with ELMo that contextualized representations from large LSTM language models transfer powerfully to downstream tasks. The Transformer generalized this insight: a single large Transformer pretrained on massive corpora, then fine-tuned on small labeled datasets, became the dominant paradigm in NLP within two years of the architecture's introduction. This pretrain–fine-tune paradigm has since extended to vision, code, biology, and chemistry.

GPT: Generative Pretraining

Radford et al. (2018) introduced GPT, a 12-layer decoder-only Transformer with 117 million parameters, pretrained on the BooksCorpus dataset (approximately 800 million words) using a standard causal language modeling objective. GPT was then fine-tuned with a task-specific classification head on nine NLP benchmarks. GPT achieved state-of-the-art results on eight of nine tasks, demonstrating that autoregressive pretraining on general text followed by minimal task-specific adaptation outperformed dedicated supervised architectures. Fine-tuned GPT achieved an accuracy of 72.8% on the MultiNLI natural language inference task, compared with the prior state of the art of 67.1%.

BERT: Bidirectional Encoders

Devlin et al. (2019) introduced BERT, arguing that the causal constraint in GPT's pretraining objective limits representation quality, since each token can only attend to preceding tokens. BERT addressed this with two novel pretraining tasks applied to a bidirectional encoder. Masked language modeling (MLM) randomly replaces 15% of input tokens with a special

[MASK] token and trains the model to predict the original tokens from fully bidirectional context. Next sentence prediction (NSP) trains the model to determine whether two input sentences are consecutive in the source document. BERT-large (340M parameters, 24 layers, 1,024-dimensional embeddings) achieved new state-of-the-art results across 11 NLP tasks upon release, including 93.2% on the Stanford Question Answering Dataset (SQuAD 1.1) compared with the prior best of 91.6%, and a GLUE benchmark score of 80.5 compared with 68.9 for the prior state of the art.

GPT-3 and Few-Shot Learning

Brown et al. (2020) introduced GPT-3, a 175-billion-parameter decoder-only Transformer trained on approximately 570 GB of text from Common Crawl, WebText2, Books1, Books2, and Wikipedia. GPT-3's most significant contribution was demonstrating striking few-shot performance: without any fine-tuning or gradient updates, GPT-3 achieves 71.8% accuracy on TriviaQA (one-shot), 79.3% on CoQA (few-shot), and 48.3% on the SuperGLUE benchmark (few-shot)—competitive with fine-tuned models of far smaller scale. Few-shot performance improved consistently with model scale, providing early evidence for the scaling law relationships formally characterized by Kaplan et al. (2020).

T5: Text-to-Text Framework

Raffel et al. (2020) introduced T5, which reframes all NLP tasks as text-to-text mappings using a unified encoder-decoder architecture. Classification becomes 'classify: positive/negative'; question answering becomes 'answer: ...' The same cross-entropy loss over generated text is applied uniformly across all tasks. T5-11B (11 billion parameters) achieved state-of-the-art results on GLUE, SuperGLUE, SQuAD, and other benchmarks at the time of publication. The T5 paper additionally provided a comprehensive empirical study of pretraining objectives, model

architectures, data sources, and training durations at scale, providing rigorous empirical guidance that shaped subsequent research directions.

Taxonomy of Transformer Model Families

The proliferation of Transformer-based models can be organized along two dimensions: (1) the architecture variant (encoder-only, decoder-only, encoder-decoder) and (2) the pretraining objective. Table 1 summarizes the primary model families.

Family	Examples	Architecture	Pretraining Objective	Primary Use
Encoder-only	BERT, RoBERTa, DeBERTa	Bidirectional encoder	Masked language modeling	Classification, NER, QA
Decoder-only	GPT, GPT-3, LLaMA, Mistral	Causal decoder	Causal language modeling	Generation, reasoning, chat
Encoder-decoder	T5, BART, mT5	Encoder + cross-attn decoder	Denoising / span masking	Translation, summarization
Multimodal encoder	CLIP, ALIGN	Dual encoder + contrastive	Contrastive image-text	Retrieval, zero-shot vision
Multimodal decoder	Flamingo, GPT-4V, LLaVA	Decoder + cross-attn vision	Causal LM with visual input	Visual QA, captioning

Note. Table 1 organizes the primary Transformer model families by architecture, pretraining objective, and primary application domain. The encoder-decoder and decoder-only families dominate current research and deployment. Closed-weight models (GPT-4, Gemini, Claude) and open-weight models (LLaMA, Mistral, OLMo, Falcon) populate the decoder-only and multimodal decoder categories; their accessibility differs substantially.

Vision Transformers

Dosovitskiy et al. (2021) demonstrated that the Transformer encoder, with minimal modification, achieves competitive or superior performance on image classification. Images are divided into non-overlapping 16 × 16 pixel patches, each linearly projected into a d-dimensional

embedding, and the resulting sequence of patch embeddings is processed by a standard Transformer encoder. A prepended [CLS] token's final representation is used for classification. The Vision Transformer (ViT) achieves 88.55% top-1 accuracy on ImageNet when pretrained on the JFT-300M dataset (300 million images), surpassing ResNet-based architectures of comparable parameter count. These results challenged the long-held assumption that convolutional inductive biases—locality and translation equivariance—are necessary for image models, establishing that Transformers can learn spatial regularities from data given sufficient pretraining.

Open-Weight Versus Closed-Weight Models

The modern LLM landscape divides into open-weight models, whose weights and (sometimes) training details are publicly released, and closed-weight proprietary models accessible only via API. LLaMA (Touvron et al., 2023), Mistral (Jiang et al., 2023), Falcon, and OLMo (Groeneveld et al., 2024) are prominent open-weight models. GPT-4, Gemini, Claude, and Grok are closed-weight models whose architecture, training data, and parameter counts are not publicly disclosed. This distinction has significant implications for scientific reproducibility: closed-weight models cannot be independently evaluated beyond API access, ablated to understand the source of capabilities, or built upon by the research community. OLMo (Groeneveld et al., 2024) represents a commitment to full transparency, releasing model weights, training data (the Pile; Gao et al., 2020), training code, and intermediate checkpoints, enabling rigorous reproducible science on foundation models.

Generative Artificial Intelligence: Diffusion Models and Multimodal Generation

From Language to Vision: The Generative Frontier

While the preceding sections trace the Transformer's development primarily through language modeling, the architecture's reach extends across modalities. Generative AI—systems that produce novel images, audio, video, and code from natural language prompts—represents perhaps the most publicly visible application of Transformer components. This section reviews two dominant generative paradigms: denoising diffusion probabilistic models and autoregressive image generation, both of which deploy Transformer-derived attention mechanisms as their core conditioning and generation engines.

Denoising Diffusion Probabilistic Models

Ho et al. (2020) introduced denoising diffusion probabilistic models (DDPMs), which learn to generate images by reversing a gradual noising process. In the forward process, Gaussian noise is progressively added to an image over T time steps until the image is indistinguishable from pure noise. The model learns the reverse process: given a noisy image at step t, predict the noise component so that the image can be denoised toward step $t - 1$. The reverse process is parameterized by a neural network—typically a UNet architecture—that takes the noisy image and the time step t as input. DDPMs achieved image quality competitive with generative adversarial networks (GANs) on CIFAR-10 and class-conditional generation on ImageNet 64×64, without the training instability that plagued GAN training.

Latent Diffusion Models and Cross-Attention Conditioning

Rombach et al. (2022) introduced latent diffusion models (LDMs), which perform the diffusion process in the compressed latent space of a pretrained variational autoencoder rather than in pixel space. This reduces the computational cost of diffusion by a factor of 8 to 16 in each spatial

dimension, enabling high-resolution (512 × 512 or 1024 × 1024) synthesis on consumer hardware. The conditioning mechanism—how a text prompt steers the generated image—is implemented through cross-attention: the UNet's intermediate feature maps serve as queries, while the text encoder (typically CLIP's or BERT's encoder) produces keys and values from the prompt embedding. At each spatial location in the UNet, cross-attention asks: given what is being generated here, which parts of the text prompt are most relevant? This mechanism directly instantiates the cross-attention formalism developed in Section 10. The Stable Diffusion model (Rombach et al., 2022) was released with open weights and has become one of the most widely deployed generative AI systems, enabling real-time text-to-image generation.

Autoregressive Image Generation

DALL-E (Ramesh et al., 2021) applied the decoder-only GPT-style autoregressive Transformer directly to image-text generation. Images are tokenized into discrete codes using a discrete variational autoencoder (dVAE), producing 32 × 32 = 1,024 image tokens. Text and image tokens are concatenated into a single sequence, and a 12-billion-parameter autoregressive Transformer predicts each token in turn. The model is trained with zero-shot generalization as the explicit goal: it achieves competitive zero-shot performance on the MS-COCO image-text matching benchmark compared with domain-specific models, demonstrating that scale and multimodal token prediction are sufficient for cross-modal understanding.

Code Generation

Codex (Chen et al., 2021) fine-tuned GPT-3 on 54 million public GitHub repositories, demonstrating that large language models can generate functionally correct code from natural language specifications. On the HumanEval benchmark—100 hand-written programming problems requiring correct Python implementations—Codex-12B solves 28.8% of problems with

a single generated sample and 72.3% when 100 samples are generated and the best is selected. Code generation models now underlie commercial programming assistants used by millions of developers daily. The architectural requirements for code differ from natural language: programs require long-range syntactic consistency (a variable defined 500 tokens earlier must be referenced consistently), formal correctness criteria, and sensitivity to tokenization artifacts on numbers and identifiers.

Multimodal Language Models

Alayrac et al. (2022) introduced Flamingo, which extends a frozen large language model with a vision encoder and cross-attention layers interleaved into the LLM, allowing the model to condition text generation on arbitrarily interleaved sequences of images and text. Flamingo achieves few-shot visual question answering performance competitive with fine-tuned models trained specifically for visual tasks, using only 4 to 32 examples in the prompt. Liu et al. (2023) introduced LLaVA, demonstrating that a vision encoder can be connected to a capable open-weight LLM (LLaMA) via a simple linear projection and instruction-tuned on visual question-answer pairs to produce a highly capable open multimodal assistant. These systems represent the current research frontier of multimodal Transformer-based generative AI.

Alignment: RLHF, Constitutional AI, and Direct Preference Optimization

The Alignment Problem

A language model pretrained on internet text optimizes the prediction of human-generated text, which includes harmful, misleading, and low-quality content alongside helpful and accurate material. Deploying such a model directly produces outputs that are frequently unhelpful, biased, or unsafe. The alignment problem concerns the gap between the optimization objective (next-token prediction) and the deployment objective (helpful, harmless, and honest behavior). Bridging this gap requires techniques that steer the model's behavior toward human-preferred outputs without degrading its capabilities.

Reinforcement Learning from Human Feedback

Ouyang et al. (2022) introduced InstructGPT, the first widely deployed RLHF-aligned language model. The RLHF pipeline consists of three stages. First, supervised fine-tuning (SFT) adapts a pretrained GPT-3 model to follow instructions using a dataset of human-written demonstrations. Second, a reward model (RM) is trained to predict human preference: human labelers compare pairs of model outputs and the RM learns to assign higher scores to preferred outputs. Third, the SFT model is further fine-tuned using Proximal Policy Optimization (PPO; Schulman et al., 2017) to maximize the reward model's score, with a KL divergence penalty that prevents the fine-tuned model from diverging too far from the SFT baseline. InstructGPT-1.3B was preferred over GPT-3-175B in 85% of evaluations, demonstrating that alignment techniques can dramatically improve perceived quality independent of raw parameter count.

RLHF has significant limitations. Reward hacking occurs when the fine-tuned model discovers outputs that score highly on the reward model without being genuinely helpful—for example, producing verbose, confident-sounding responses that satisfy superficial markers of

quality identified by labelers (Gao et al., 2023). The reward model itself may be miscalibrated if the human labeler pool is unrepresentative, if labelers cannot assess technical correctness, or if the preference data distribution does not cover the full range of deployment contexts. Casper et al. (2023) provide a comprehensive review of RLHF's empirical failure modes.

Constitutional AI

Bai et al. (2022) proposed Constitutional AI (CAI) as an alternative approach to alignment that reduces reliance on human preference labeling. A 'constitution'—a set of principles expressing the desired behavior (e.g., 'be helpful, harmless, and honest')—is used to generate AI feedback: the model critiques its own outputs with reference to constitutional principles and revises them accordingly. A reinforcement learning stage then trains a preference model on this AI-generated feedback (AI feedback, or AIFF) rather than human feedback. CAI scales more readily than RLHF because human labeling is required only to write the constitution and to validate outputs, not to generate preference comparisons for the full training distribution. Constitutional AI underlies the Claude family of models (Anthropic, 2023).

Direct Preference Optimization

Rafailov et al. (2023) introduced Direct Preference Optimization (DPO), which reformulates the RLHF objective as a supervised learning problem, eliminating the need for a separate reward model and the reinforcement learning stage entirely. The key insight is that the optimal policy under the KL-penalized RLHF objective can be expressed in closed form as a function of the ratio between the policy and the reference (SFT) model's probabilities. This allows the RLHF objective to be reparametrized directly in terms of the policy, yielding a simple binary cross-entropy loss over preferred and dispreferred responses:

$$L_{DPO}(\pi_\theta) = -E[(y_w, y_l) \sim D] [\log \sigma(\beta \log \pi_\theta(y_w|x)/\pi_{ref}(y_w|x) - \beta \log \pi_\theta(y_l|x)/\pi_{ref}(y_l|x))]$$

where y_w denotes the winning (preferred) response, y_l the losing (dispreferred) response, x the prompt, β a temperature parameter, and π_ref the reference policy. DPO achieves alignment quality competitive with PPO-based RLHF while requiring substantially less computational infrastructure. Its simplicity has driven rapid adoption; DPO or DPO variants are used in the alignment stage of numerous open-weight models including Mistral-Instruct and LLaMA-2-Chat.

Theoretical Expressivity of Transformers

The theoretical properties of Transformer-based architectures span a spectrum of assumptions, from idealized constructions with infinite precision and unbounded depth to practical finite models with floating-point arithmetic. It is essential to distinguish results in each category to avoid conflating theoretical possibility with practical capability.

Universal Approximation: Results for Ideal Transformers

The classical universal approximation theorem (Cybenko, 1989; Funahashi, 1989; Hornik, 1991) establishes that a feedforward network with a single hidden layer and a nonlinear activation can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary precision, given sufficient width. Note that these three results were established concurrently and independently; all three merit attribution.

Yun et al. (2020) proved an analogous result for Transformers: a Transformer with multi-head self-attention and feedforward sublayers is a universal approximator of sequence-to-sequence functions. More precisely, any continuous permutation-equivariant function mapping sequences of fixed length n from a compact domain in $\mathbb{R}^d$ to $\mathbb{R}^d$ can be approximated to arbitrary precision by a sufficiently deep and wide Transformer. This result requires explicit qualification: it applies to functions of fixed sequence length; it may require depth proportional to the approximation error; and it is an existence result—it guarantees that the approximating Transformer exists but says nothing about whether gradient descent on finite data will find it. The practical distance between 'a Transformer can represent this function' and 'training a Transformer will learn this function' is substantial and not addressed by the approximation theorem.

Turing Completeness: Results for Idealized Transformers with Infinite Precision

Pérez et al. (2021) proved that a Transformer with arbitrary-precision arithmetic (no floating-point rounding) and positional encodings that uniquely identify each position can simulate any Turing machine. The simulation encodes the tape content in token embeddings, uses positional encodings as tape cell indices, exploits self-attention for content-addressable memory reads, and uses feedforward networks to implement the state transition function. This result establishes theoretical computational universality under idealized conditions.

These ideal conditions are not approximated by practical models. Floating-point arithmetic introduces rounding errors that accumulate across layers. Context windows are finite, limiting the 'tape' to a fixed length. Depth is finite, bounding the number of computational steps. Weiss et al. (2021) formalized the gap between theory and practice by proving that Transformers with finite precision are equivalent to RASP (Restricted Access Sequence Processing) machines—a restricted class of automata strictly weaker than full Turing machines, equivalent in power to counter machines. This result is important for setting realistic expectations: practical Transformers are not Turing complete and cannot compute arbitrary functions given sufficient prompting.

Hierarchical Limitations of Bounded-Norm Transformers

Hahn (2020) proved a third category of result applying to bounded-norm Transformers: Transformers with bounded weight norms cannot recognize certain hierarchically structured formal languages, specifically those requiring resolution of arbitrarily deep nested structure such as balanced parentheses. This limitation is independent of depth and width; it is a consequence of the attention mechanism's fundamental inability to count to arbitrary depth with bounded precision. This theoretical finding has practical implications for formal reasoning and program verification: certain recursive structures require either external memory augmentation or explicit positional counting mechanisms beyond standard attention.

Summary: Three Tiers of Expressivity

In summary, Transformer expressivity should be understood at three levels: (1) Ideal Transformers with infinite precision and depth are Turing complete universal approximators. (2) Bounded-norm Transformers with finite precision approximate a restricted class of automata that cannot resolve arbitrary hierarchical structure. (3) Practical trained Transformers, evaluated empirically, exhibit surprising capability within these bounds, including sophisticated reasoning, code generation, and mathematical problem-solving, though they exhibit systematic failures on tasks requiring strict compositional generalization or deep recursion. Conflating these three tiers produces both overconfident and underconfident claims about what Transformers can do.

Scaling Laws: Empirical Regularities and Their Limits

The Kaplan et al. Power Laws

Kaplan et al. (2020) conducted a systematic study of how language model performance scales with three variables: parameter count N, training data tokens D, and compute budget C = 6ND (a common approximation for Transformer training FLOPs). They found that cross-entropy loss follows smooth power-law relationships:

$$L(N) \approx (N_c / N)^{\alpha_N}, \quad L(D) \approx (D_c / D)^{\alpha_D}, \quad L(C) \approx (C_c / C)^{\alpha_C}$$

with empirically measured exponents $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $\alpha_C \approx 0.050$ for large autoregressive Transformer language models. These power laws held across more than six orders of magnitude in parameter count and three orders of magnitude in compute, suggesting a structural regularity in how Transformer-based language models learn.

Chinchilla Scaling and the Data-Optimal Allocation

Hoffmann et al. (2022) revisited the Kaplan et al. findings with a more comprehensive experimental design in which both model size and training duration were varied independently. They found that the Kaplan et al. recommendations substantially under-allocated training data relative to model size: for a fixed compute budget C, the optimal allocation requires training a model of N = C / (6D) parameters for D = C / (6N) tokens, where N and D should scale in approximately equal proportion. A compute-optimal (Chinchilla) model with 70 billion parameters should be trained on approximately 1.4 trillion tokens—far more than the roughly 300 billion tokens used for comparably sized models following Kaplan et al.'s guidance. The Chinchilla model (70B parameters, 1.4T tokens) outperformed Gopher (280B parameters, 300B tokens) on almost all downstream benchmarks despite having four times fewer parameters, demonstrating that data efficiency matters as much as parameter count.

An important qualifier not captured in the Chinchilla formulation: the compute-optimal training budget minimizes loss for a given training compute expenditure but does not account for inference costs. In practice, a smaller model trained on more tokens is cheaper to serve at scale because each forward pass requires fewer FLOPs. This inference-cost consideration has led many frontier model developers to deliberately deviate from Chinchilla optimality, training smaller models for substantially more tokens than Chinchilla recommends. LLaMA (Touvron et al., 2023) trained a 7-billion-parameter model on 1 trillion tokens—far beyond Chinchilla optimality for a compute-limited regime—specifically to optimize inference efficiency.

Epistemic Limits of Scaling Laws

The scaling law framework should be understood as an empirical regularity in a specific training regime, not as a fundamental law of nature. Several important caveats apply. First, the power-law exponents vary across domains: vision models, code models, and multilingual models exhibit different scaling behavior than English text models. Second, data quality strongly affects the effective scaling rate in ways not captured by simply counting training tokens. Third, the laws were derived for a specific Transformer architecture family and may not generalize to architectures with different inductive biases. Fourth, and most fundamentally, the power laws describe loss—they do not directly predict capability on specific downstream tasks, which may exhibit more discontinuous or non-monotonic relationships with scale.

Emergent Abilities: Observations, Critiques, and the Current Consensus

The Original Observations

Wei et al. (2022b) documented what they termed emergent abilities of large language models: capabilities that appeared to arise abruptly as model scale crossed certain thresholds. In their framing, below a threshold model scale, performance on a task is near chance; above it, performance is substantially and discontinuously above chance. Examples included multi-digit arithmetic, multi-step word problems, temporal sequence reasoning, and analogical reasoning. The apparent abruptness distinguished emergence from the smooth power-law improvement predicted by scaling laws: emergent abilities seemed to appear from nothing rather than improving gradually.

The Metric-Artifact Critique

Schaeffer et al. (2023) provided a substantial reexamination of the emergence claim, demonstrating that the observed discontinuities are in most cases an artifact of the evaluation metric rather than a genuine property of the underlying model capability. When tasks are evaluated with nonlinear or threshold metrics—such as binary correct/incorrect—smooth improvement in the model's log-probability of the correct answer maps to a discontinuous improvement in the accuracy metric, creating an apparent phase transition. When continuous metrics such as log-probability or token-level accuracy are substituted, the apparent emergent behavior smooths out into the continuous power-law improvement predicted by scaling laws. Schaeffer et al. demonstrated this empirically across the majority of tasks reported as emergent by Wei et al. (2022b).

This critique does not imply that all capability improvements are smooth and gradual. Some tasks—particularly those involving compositional reasoning where multiple independent steps must all be correct—may exhibit genuine nonlinearities from the combination of several

continuously improving component capabilities. However, these cases can generally be explained without invoking a qualitatively new phenomenon of emergence. The current consensus in the research community, as of early 2026, is that the strong version of the emergence hypothesis—that capabilities arise discontinuously from nothing above a threshold—is not well supported empirically, while the weaker version—that performance on some complex tasks improves faster than linear in scale, and that new capabilities become practically useful at specific scale thresholds—remains descriptively accurate and practically important.

Chain-of-Thought Prompting

Wei et al. (2022a) demonstrated that prompting large language models to generate explicit intermediate reasoning steps—chain-of-thought prompting—dramatically improves performance on multi-step reasoning tasks. On the GSM8K grade school mathematics benchmark, chain-of-thought prompting improves PaLM-540B from 17.9% to 56.9% accuracy; on MATH (harder competition-level problems), chain-of-thought improves performance from near chance to approximately 50% for the strongest models. The mechanism is interpretable: generating intermediate steps creates tokens that subsequent attention can refer to, extending the model's effective working memory from the parameter space to the context window. Chain-of-thought benefits are reliably observed above approximately 100 billion parameters in the original study, though subsequent work with instruction-tuned smaller models has achieved chain-of-thought benefits at lower scales. Whether chain-of-thought constitutes genuine reasoning or a sophisticated simulation of reasoning—producing correct answers through a learned template of intermediate steps rather than through genuine inference—remains an open and contested question (Marcus, 2022).

In-Context Learning as Implicit Bayesian Inference

The Phenomenon

In-context learning (ICL) is the ability of a large language model to adapt to a new task from examples provided in the prompt, without any update to the model's parameters. Given a few examples of English-to-French translation in the prompt, a model that was never explicitly fine-tuned for translation performs translation on the next example. This parameter-free adaptation is qualitatively distinct from standard machine learning, which requires gradient descent for adaptation, and suggests that the model has internalized a general learning procedure during pretraining.

The Bayesian Interpretation

Xie et al. (2022) provided a theoretical account of ICL as implicit Bayesian inference. Under a data-generating model in which training documents are drawn from a mixture of latent tasks (e.g., translation, summarization, QA), the model's pretraining objective can be shown to implicitly learn the mixture weights as a prior P(task). At test time, the prompt examples constitute observations from which the model can infer the posterior distribution over tasks:

$$P(\text{task} \mid \text{prompt examples}) \propto P(\text{prompt examples} \mid \text{task}) \cdot P(\text{task})$$

Xie et al. (2022) proved that under this model, a Transformer trained on such a mixture provably implements Bayesian inference at test time: the attention mechanism integrates the evidence from prompt examples to update the posterior over the latent task, and the model's next-token predictions are conditioned on this posterior. This interpretation provides a mechanistic explanation for ICL grounded in probability theory rather than in unexplained scaling phenomena.

Gradient Descent in Context

Akyürek et al. (2022) demonstrated a complementary result: Transformers can implement gradient descent over linear models in their forward pass. Given examples from a linear regression task in the context, the Transformer's output matches the prediction of a linear model trained by gradient descent on exactly those examples. This shows that the attention mechanism can implement a learned optimization algorithm—not merely pattern matching over surface features—providing a concrete mechanistic account of how adaptation occurs within the forward pass without gradient updates to model parameters. Together with the Bayesian interpretation, this result suggests that ICL engages sophisticated internal mechanisms rather than simple template matching.

Mechanistic Interpretability of Attention

Motivation

As Transformer-based models exhibit increasingly powerful and sometimes unexpected capabilities, understanding their internal mechanisms becomes essential both for scientific understanding and for safety-critical deployment. Mechanistic interpretability aims to identify the specific computational circuits within trained models that are causally responsible for particular behaviors—not merely to describe input-output behavior but to reverse-engineer the internal algorithms that produce it.

Induction Heads

Olsson et al. (2022) identified the induction head: a two-head circuit that implements a general copy-and-predict algorithm. The circuit comprises two attention heads acting in sequence. A 'previous token head' at an earlier layer creates a representation of each token that incorporates information from the immediately preceding token. An 'induction head' proper at a later layer attends backward to find prior occurrences of the current token and predicts the token that followed each prior occurrence. This implements the general rule: if A was followed by B before, predict B when A recurs. Olsson et al. (2022) demonstrated that the formation of induction heads during training correlates with a phase transition in in-context learning performance—a sudden improvement in ICL that aligns temporally with the emergence of induction circuits in the residual stream. This provides one of the clearest causal links between an identified internal circuit and an observable capability.

Indirect Object Identification Circuit

Wang et al. (2022) identified a more complex circuit in GPT-2-small responsible for indirect object identification (IOI): completing sentences such as 'John gave Mary the gift. Mary

thanked ___' with 'John.' Through a combination of activation patching, attention knockout, and path ablation, Wang et al. traced the mechanism to approximately 26 attention heads organized into three functional groups: name mover heads that copy the indirect object name to the output position, duplicate token heads that suppress the direct object name (Mary), and S-inhibition heads that regulate the competition between the two candidate names. The existence of such specific, modular circuits challenges the view of neural networks as undifferentiated statistical averages and supports a hybrid account in which connectionist learning discovers symbolic-like computational modules.

Analytical Methods

The mechanistic interpretability toolkit includes attention visualization—plotting the attention weight matrix α_{ij} as a heatmap; activation patching (causal tracing), which replaces an activation at a specific layer and position with a corrupted or clean value and measures the effect on model output; circuit ablation, which systematically sets heads or components to their mean activation and measures performance impact on targeted tasks; and probing classifiers, which train linear models on internal representations to test whether specific features are linearly decodable. Each method has limitations—attention weights do not uniquely determine information flow due to the value transformation—and the field increasingly combines multiple methods for robust causal attribution.

Philosophical Implications

The mechanistic interpretability findings suggest that attention heads implement specific structured algorithms—copy-and-predict, coreference resolution, syntactic agreement checking—as emergent behaviors of gradient descent over continuous representations. This partially bridges the classical dichotomy between connectionist and symbolic AI: gradient-trained networks

discover algorithms recognizable as symbolic operations. However, the bridge is partial, and the claim that these circuits constitute genuine symbolic computation in the sense of Fodor and Pylyshyn (1988)—systematic, productive, compositional—remains contested. The circuits discovered thus far are modular within their trained distribution but show limited ability to compose systematically to solve novel tasks that require applying learned rules in new combinations. This limitation connects mechanistic interpretability to the broader challenge of systematic generalization discussed in the following section.

Limitations, Failure Modes, and Future Directions

Quadratic Complexity and Efficient Attention

The $O(n^2)$ computational complexity of full self-attention with respect to sequence length remains the binding constraint for long-context applications. FlashAttention (Dao et al., 2022) reduces memory footprint and wall-clock time through hardware-aware tiling but does not reduce the fundamental algorithmic complexity. Sparse attention variants (Sections 12–13) reduce FLOPs but introduce approximation error or require fixed sparsity patterns that may not match the input's relational structure. State-space models (Gu & Dao, 2023) achieve linear complexity but trade approximation quality for efficiency. No current method achieves the full accuracy of softmax attention at linear cost for general inputs; this remains an open theoretical problem.

Hallucination and Factual Reliability

Language models frequently generate confident but factually incorrect statements—a phenomenon termed hallucination. The root cause is architectural: next-token prediction training rewards plausible continuations of text, not factually accurate ones. The training corpus contains both true and false statements; the model learns to reproduce the distribution of both. Retrieval-augmented generation (RAG) partially mitigates hallucination by providing the model with retrieved factual context at inference time, but introduces new failure modes: the model must decide when to retrieve, and must evaluate whether retrieved content is trustworthy and relevant. Formal verification integration—connecting LLM generation to symbolic verifiers that check mathematical or logical correctness—represents an active research direction but remains limited to narrow formal domains.

Long-Context Faithfulness: The Lost-in-the-Middle Problem

Liu et al. (2023b) documented a systematic failure mode distinct from context window length: language models exhibit significantly lower accuracy on information presented in the middle of long contexts compared with information at the beginning or end. On a multi-document question answering task with 20 documents in the context, model accuracy on relevant documents near the middle of the context was approximately 30–40 percentage points lower than on documents at the beginning or end. This 'lost-in-the-middle' phenomenon persists across models including GPT-3.5-Turbo and Claude, and does not improve substantially with longer context windows. The mechanism is hypothesized to relate to the U-shaped attention pattern in which models attend most strongly to the beginning of the context (due to the causal masking structure placing these tokens in all subsequent attention computations) and to the end (recency effects in autoregressive generation). Long-context faithfulness represents a significant practical limitation for document analysis, legal review, and scientific literature synthesis applications.

Systematic Generalization and Compositional Reasoning

Lake and Baroni (2018) demonstrated that sequence-to-sequence RNNs fail to systematically generalize—applying learned rules to novel combinations of primitive concepts—on the SCAN benchmark. Hahn (2020) established that Transformers share this limitation at the formal level: bounded-norm Transformers cannot recognize certain hierarchically composed formal languages. Empirically, frontier LLMs exhibit impressive performance on many reasoning benchmarks but continue to fail on tasks requiring strict compositional reasoning, particularly those involving novel object-attribute bindings, deeply nested recursive structures, or multi-step mathematical derivations outside their training distribution. These failures suggest that current Transformers learn statistical correlations over compositional patterns rather than implementing the systematic, productive generalization characteristic of human language and thought.

Energy, Compute, and Sustainability

Training GPT-3 required an estimated 3.14×10^{23} floating-point operations, corresponding to hundreds of megawatt-hours of electricity and a carbon footprint comparable to hundreds of transatlantic flights. Inference costs at deployment scale are comparable or greater: serving millions of queries per day to a 70-billion-parameter model requires thousands of GPUs running continuously. These costs are driving active research into model compression—quantization (reducing numerical precision of weights from 16 to 4 bits or below), pruning (removing low-importance weights), and knowledge distillation (training smaller student models to mimic larger teacher models; Hinton et al., 2015). Mixtral (Jiang et al., 2024) demonstrates that mixture-of-experts architectures—activating only a sparse subset of model parameters for each token—can achieve performance competitive with dense models at a fraction of the inference compute cost.

Future Directions

The most consequential open research directions include: (1) efficient attention algorithms that achieve linear complexity without sacrificing full-attention accuracy; (2) architectural innovations for systematic compositional generalization, potentially combining differentiable neural computation with explicit symbolic reasoning or program synthesis; (3) alignment techniques that scale to frontier model capabilities without reward hacking and that provide formal guarantees rather than empirical demonstrations; (4) multimodal grounding that connects language model representations to perceptual reality, reducing hallucination through direct supervision from the physical world; and (5) interpretability methods that scale from understanding individual circuits in small models to understanding the distributed computations of hundred-billion-parameter systems. The Transformer's current dominance reflects genuine architectural advantages—scalability, parallelism, relational computation—but is not a permanent endpoint.

The limitations documented in this section point clearly toward the innovations the next architectural generation will need to achieve.

Conclusion

This systematic review has traced the intellectual genealogy of deep learning architectures from convolutional networks through recurrent systems, gated memory, attention mechanisms, and the Transformer to the current era of large-scale foundation models and generative artificial intelligence. Each architectural advance was motivated by a specific limitation of its predecessor: CNNs resolved the parameter inefficiency and spatial invariance failures of feedforward networks; RNNs introduced temporal memory; LSTMs resolved the vanishing gradient problem through gated additive cell state updates that provide a clean gradient highway; and the Transformer resolved the sequential bottleneck through fully parallel self-attention that eliminates recurrence while preserving—and indeed enhancing—the capacity for long-range dependency modeling.

The attention mechanism is the linchpin of this progression. By reformulating information access as a dynamic, content-dependent retrieval operation—grounded in the query-key-value abstraction whose intellectual roots span cognitive neuroscience, the Neural Turing Machine, Memory Networks, and neural machine translation—attention enables models to selectively retrieve exactly what is needed from any position in the input. The full taxonomy of attention variants—self-attention, multi-head attention, cross-attention, masked attention, local and global attention, sparse attention, Flash Attention with its hardware-aware tiling and online softmax, linear attention with its O(1)-per-step inference reformulation, and rotary positional embeddings with their relative-position invariance—represents an expanding toolkit whose continued development reflects the community's sustained effort to overcome the quadratic complexity barrier while preserving expressive power.

The Transformer's theoretical properties span three tiers that must be clearly distinguished: ideal Transformers with infinite precision are universal approximators and Turing complete;

bounded-norm Transformers with finite precision are equivalent to counter machines and cannot resolve arbitrary hierarchical structure; and practical trained Transformers exhibit impressive but imperfect capabilities that are well described by neither theoretical extreme. Scaling laws reveal predictable power-law relationships between scale and loss, with optimal data-to-parameter allocation specified by Chinchilla scaling and practical inference-cost considerations creating rational deviations from training optimality. The contested phenomenon of emergent abilities is more accurately described as metric-dependent observation of continuous capability improvement, with genuine nonlinearities arising from task complexity rather than architectural phase transitions.

In-context learning represents one of the most scientifically striking capabilities to emerge from scale: the ability to perform Bayesian inference over task distributions and to implement gradient-descent-like optimization within the forward pass, without any parameter updates. Mechanistic interpretability reveals that these capabilities are supported by identifiable internal circuits—induction heads, name movers, S-inhibition heads—that implement recognizable algorithms as emergent behaviors of gradient descent. The generative AI frontier extends the Transformer's reach to image synthesis through diffusion models conditioned by cross-attention, to code generation, to multimodal understanding, and to aligned, instruction-following assistants through RLHF, Constitutional AI, and direct preference optimization.

Significant limitations remain: hallucination from the absence of truth grounding in next-token prediction training; long-context faithfulness failures manifested as systematic neglect of middle-context information; inability to achieve systematic compositional generalization; quadratic complexity in long sequences; and the environmental and economic costs of large-scale training and inference. These limitations define the research agenda for the coming decade.

For a reader new to this field, the essential conceptual progression is as follows: feedforward networks provide the basic computational substrate; CNNs add spatial inductive bias through weight sharing; RNNs add temporal memory through hidden state propagation; LSTMs add gated long-term memory through additive cell state updates; attention adds dynamic content-based retrieval that decouples memory access from sequential computation; the Transformer generalizes attention into a fully parallel relational computation engine; foundation models demonstrate that scaling this engine over vast and diverse data produces capabilities that approach and in specific domains exceed human expert performance; and generative models apply these capabilities to produce novel images, code, and other artifacts from natural language specifications. The Transformer is not merely an architecture. It is a computational paradigm—a shift from positional, sequential computation to relational, parallel computation over learned geometric representations. Understanding this paradigm in its full depth, with its mathematical foundations, its variants, its theoretical scope and limits, its empirical properties, and its societal implications, is the foundational requirement for rigorous and responsible work in artificial intelligence.

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., & Zhou, D. (2022). What learning algorithm is in-context learning? Investigations with linear models. International Conference on Learning Representations.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J. L., Borgeaud, S., … Simonyan, K. (2022). Flamingo: A visual language model for few-shot learning. Advances in Neural Information Processing Systems, 35, 23716–23736.

Anthropic. (2023). Claude's constitution [Technical report]. Anthropic.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. International Conference on Learning Representations.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., … Kaplan, J. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv preprint arXiv:2204.05862.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166. https://doi.org/10.1109/72.279181

Bloc97. (2023). NTK-aware scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation [Online forum post]. Reddit. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., Wang, T., Marks, S., Segerie, C.-R., Carroll, M., Peng, A., Christoffersen, P., Damani, M., Slocum, S., Anwar, U., … Hadfield-Menell, D. (2023). Open problems and fundamental limitations of reinforcement learning from human feedback. Transactions on Machine Learning Research.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1724–1734. https://doi.org/10.3115/v1/D14-1179

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking attention with Performers. International Conference on Learning Representations.

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. Proceedings of the 2019 ACL Workshop BlackboxNLP, 276–286. https://doi.org/10.18653/v1/W19-4828

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems, 2(4), 303–314. https://doi.org/10.1007/BF02551274

Dao, T. (2023). FlashAttention-2: Faster attention with better parallelism and work partitioning. International Conference on Learning Representations.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. Advances in Neural Information Processing Systems, 35, 16344–16359.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. International Conference on Learning Representations.

Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. Neural Networks, 2(3), 183–192. https://doi.org/10.1016/0893-6080(89)90003-8

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., & Leahy, C. (2020). The Pile: An 800GB dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027.

Gao, L., Schulman, J., & Hilton, J. (2023). Scaling laws for reward model overoptimization. Proceedings of the International Conference on Machine Learning, 10835–10866.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. Proceedings of the International Conference on Machine Learning, 1243–1252.

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing machines. arXiv preprint arXiv:1410.5401.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., … Smith, N. A. (2024). OLMo: Accelerating the science of language models. arXiv preprint arXiv:2402.00838.

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

Gu, A., Goel, K., & Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. International Conference on Learning Representations.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. Transactions of the Association for Computational Linguistics, 8, 156–171. https://doi.org/10.1162/tacl_a_00306

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33, 6840–6851.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Davison, I., Aslanides, J., Casas, G. A., Ring, R., Millican, K., Simonyan, K., Green, L., … Sifre, L.

(2022). Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of Physiology, 160(1), 106–154. https://doi.org/10.1113/jphysiol.1962.sp006837

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proceedings of the International Conference on Machine Learning, 448–456.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., & El Sayed, W. (2023). Mistral 7B. arXiv preprint arXiv:2310.06825.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, M., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., … El Sayed, W. (2024). Mixtral of experts. arXiv preprint arXiv:2401.04088.

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach [Technical report]. Institute for Cognitive Science, University of California, San Diego.

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

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. Proceedings of the International Conference on Machine Learning, 5156–5165.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.

Kitaev, N., Kaiser, L., & Levskaya, A. (2020). Reformer: The efficient transformer. International Conference on Learning Representations.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 66–71. https://doi.org/10.18653/v1/D18-2012

Lake, B. M., & Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. Proceedings of the International Conference on Machine Learning, 2879–2888.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). Visual instruction tuning. Advances in Neural Information Processing Systems, 36.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023b). Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12, 157–173. https://doi.org/10.1162/tacl_a_00638

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 1412–1421. https://doi.org/10.18653/v1/D15-1166

Marcus, G. (2022). Deep learning is hitting a wall. Nautilus. https://nautil.us/deep-learning-is-hitting-a-wall-238440/

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133. https://doi.org/10.1007/BF02478259

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. Proceedings of the 11th Annual Conference of the International Speech Communication Association, 1045–1048.

Milakov, M., & Gimelshein, N. (2018). Online normalizer calculation for softmax. arXiv preprint arXiv:1805.02867.

Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent models of visual attention. Advances in Neural Information Processing Systems, 27, 2204–2212.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., … Clark, J. (2022). In-context learning and induction heads. Transformer Circuits Thread.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. Proceedings of the International Conference on Machine Learning, 1310–1318.

Peng, B., Quesnelle, J., Fan, H., & Shippole, E. (2023). YaRN: Efficient context window extension of large language models. International Conference on Learning Representations.

Pérez, J., Marinković, J., & Barceló, P. (2021). Attention is Turing complete. Journal of Machine Learning Research, 22(75), 1–35.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, 2227–2237. https://doi.org/10.18653/v1/N18-1202

Press, O., Smith, N. A., & Lewis, M. (2022). Train short, test long: Attention with linear biases enables input length extrapolation. International Conference on Learning Representations.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. Proceedings of the International Conference on Machine Learning, 8748–8763.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training [Technical report]. OpenAI.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems, 36.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1–67.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. Proceedings of the International Conference on Machine Learning, 8821–8831.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10684–10695. https://doi.org/10.1109/CVPR52688.2022.01042

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

Schaeffer, R., Miranda, B., & Koyejo, S. (2023). Are emergent abilities of large language models a mirage? Advances in Neural Information Processing Systems, 36.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Schuster, M., & Nakamura, K. (2012). Japanese and Korean voice search. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, 5149–5152. https://doi.org/10.1109/ICASSP.2012.6289079

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 1715–1725. https://doi.org/10.18653/v1/P16-1162

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, 464–468. https://doi.org/10.18653/v1/N18-2074

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56), 1929–1958.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, 27, 3104–3112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2021). Efficient transformers: A survey. ACM Computing Surveys, 55(6), 1–28. https://doi.org/10.1145/3530811

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. Cognitive Psychology, 12(1), 97–136. https://doi.org/10.1016/0010-0285(80)90005-5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 5797–5808. https://doi.org/10.18653/v1/P19-1580

Waleffe, R., Byeon, W., Massaroli, S., Sun, B., Bhatia, K., Singh, S., Dao, T., Keutzer, K., & Ré, C. (2024). An empirical study of Mamba-based language models. arXiv preprint arXiv:2406.07887.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., & Steinhardt, J. (2022). Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. International Conference on Learning Representations.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., & Zhou, D. (2022a). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824–24837.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022b). Emergent abilities of large language models. Transactions on Machine Learning Research.

Weiss, G., Goldberg, Y., & Yahav, E. (2021). Thinking like Transformers. Proceedings of the International Conference on Machine Learning, 11080–11090.

Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. International Conference on Learning Representations.

Xie, S. M., Raghunathan, A., Liang, P., & Ma, T. (2022). An explanation of in-context learning as implicit Bayesian inference. International Conference on Learning Representations.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., & Kumar, S. (2020). Are transformers universal approximators of sequence-to-sequence functions? International Conference on Learning Representations.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 33, 17283–17297.