AI Demystified for CXO

Jaideep Ganguly, Sc.D. (MIT)
https://jganguly.github.io

23 Jan 2026

About the Speaker

- 35 years of industry experience in India and USA.
- Technical Fellow & Consultant
  - Quantum Computing Algorithms in Reinforced Learning
  - AI in Medicine
- Amazon, Technology Leader
- Microsoft, Technology Leader
- Massachusetts Institute of Technology (MIT),
  Doctor of Science, Master of Science
- Indian Institute of Technology, Kharagpur,
  B.Tech (Hons.)

Outline

What is Artificial Intelligence (AI)?

- **Artificial Intelligence (AI)** refers to systems that perform tasks which typically require human intelligence. These tasks include:
  - Perception (vision, speech)
  - Reasoning and decision-making
  - Learning from data
  - Problem solving and planning
- Core paradigms of AI
  - **Symbolic AI**: Rules, logic, knowledge graphs
  - **Statistical AI**: Probabilities, inference, optimization
  - **Neural AI**: Learning representations from data
- Modern AI systems are primarily based on:
  - statistical learning
  - optimization
  - large-scale computation
- AI does *not* imply consciousness or general intelligence; most systems are task-specific.

History & Evolution of AI

- **Foundations (1950s–1960s)**
  - Alan Turing (University of Manchester): Turing Test, computability
  - John McCarthy (MIT / Stanford): coined the term "Artificial Intelligence"
  - Marvin Minsky (MIT): symbolic AI, early cognitive architectures
- **Expert Systems Era (1970s–1980s)**
  - Edward Feigenbaum (Stanford University): knowledge-based systems
  - MYCIN and DENDRAL expert systems
  - Patrick Winston (MIT): knowledge representation, AI pedagogy
- **Statistical Machine Learning (1990s–2000s)**
  - Vladimir Vapnik (AT&T Bell Labs): Support Vector Machines
  - Judea Pearl (UCLA): probabilistic reasoning, Bayesian networks
  - Growing influence of MIT in robotics and learning-based AI
- **Deep Learning Revolution (2010s)**
  - Geoffrey Hinton (University of Toronto / Google): Deep Neural Networks
  - Yann LeCun (AT&T Bell Labs / Meta): convolutional neural networks
  - Yoshua Bengio (Université de Montréal): representation learning
  - MIT CSAIL: advances in robotics, perception, and human-centered AI
- **Foundation Models (2020s–Present)**
  - Transformer architecture (Google Brain)
  - Large-scale models developed at OpenAI, DeepMind, Meta
  - Ongoing contributions from MIT in AI safety, systems, and applied AI

AI, ML, Deep Learning, LLM, Generative AI

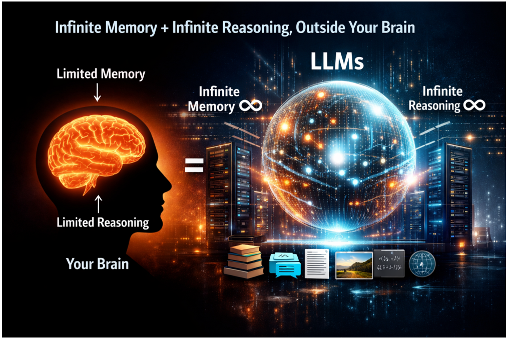
- **Artificial Intelligence (AI)**
  - Broad goal: systems that exhibit human-like intelligence
  - Includes reasoning, planning, perception, and decision-making
- **Machine Learning (ML)**
  - Subset of AI focused on learning patterns from data. Improves performance without explicit rule-based programming
  - Types of ML
    - **Supervised Learning**: Labeled data (prediction, classification)
    - **Unsupervised Learning**: Structure discovery (clustering)
    - **Reinforcement Learning**: Learning via feedback and rewards
- **Deep Learning (DL)**
  - Subset of ML using multi-layer neural networks
  - Excels at vision, speech, language, and complex representations
- **LLMs (Large Language Models)** are a specific type of model that generate text and language-based outputs. All LLMs are Generative AI
- **Generative AI (GenAI)**
  - Subset of DL focused on generating new content; Produces text, images, code, audio, and simulations.
  - Not all Generative AI are LLMs.

Each layer is a specialization of the one above it

Why AI Matters Now

- AI is not new — but **three forces converged** in the last decade:
  - Explosion of digital data
  - Massive, affordable compute (cloud, GPUs)
  - Breakthrough algorithms (deep learning, transformers)
- AI has crossed a threshold:
  - From research labs to real business deployment
  - From niche tools to enterprise platforms
- Generative AI made AI *visible* and accessible to everyone

Key message: AI is now a competitive necessity, not an experiment.



The AI Stack: A CXO View

- **Data Layer**
  - Enterprise data, quality, ownership, governance
- **Model Layer**
  - Classical ML, deep learning, foundation models
- **Application Layer**
  - Copilots, automation, decision-support systems
- **Infrastructure Layer**
  - Cloud, on-prem, security, cost management

Key message: AI value depends on the *entire stack*, not just models.

Where AI Creates Business Value

- **Revenue Growth**
  - Personalization
  - pricing optimization
  - sales effectiveness
- **Cost Reduction**
  - Automation
  - Process efficiency
  - Reduced manual effort
- **Risk Reduction**
  - Fraud detection
  - Compliance
  - Anomaly detection
- **Decision Quality & Speed**
  - Forecasting
  - Scenario analysis
  - Recommendations

Key message: AI is a business lever, not an IT project.

AI Use Cases: What Works, What Doesn't

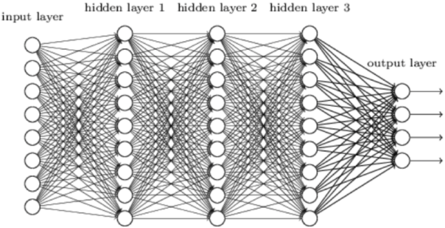
- **High-ROI Use Cases**
  - Demand forecasting
  - Recommendations and personalization
  - Document and knowledge automation
  - Customer support copilots
- **High-Risk / Overhyped Use Cases**
  - Fully autonomous decision-making
  - Replacing judgment-heavy roles
  - AI without clean or owned data
- **Why Pilots Fail**
  - Unclear business ownership
  - Poor data foundations
  - No success metrics

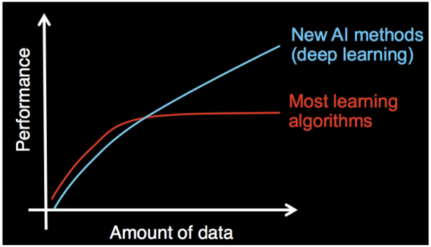
Key message: Successful AI starts with the right problem, not the right model.

Core AI Technologies

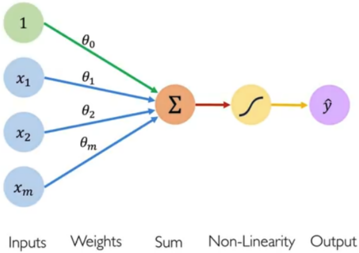
1. Deep Neural Network
2. Generative AI (Text, Image, Audio, Video)
3. Large Language Models (LLMs)
4. Multimodal AI
5. Reinforcement Learning (RL)
6. AI Agents & Autonomy
7. Explainable AI (XAI)

Deep Neural Network





Activation



Output

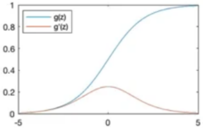
Linear combination of inputs

$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i\,\theta_i\right)$$

Non-linear activation function

Bias

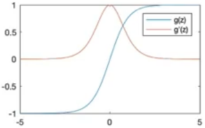
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

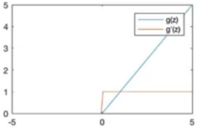
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

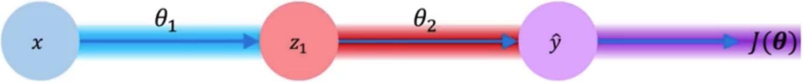
Rectified Linear Unit (ReLU)



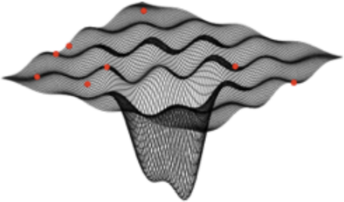
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Back Propagation



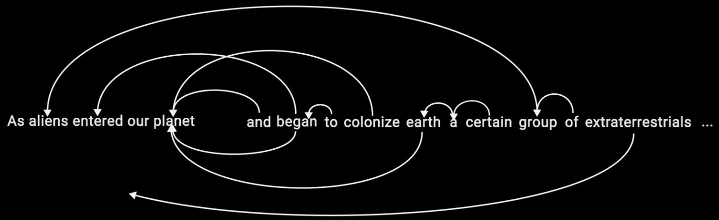
$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial \theta_1}$$



LLM - Large Language Model

1. **LLM:** An LLM learns patterns from massive text datasets and uses them to generate or interpret language. It relies on a transformer architecture with self-attention.
2. **Transformer:** A type of neural network used by modern AI models (like GPT, BERT, Claude). Its core innovation is the self-attention mechanism.
3. **Self-attention:** The model understands the relationships between words in a sentence all at the same time, rather than sequentially.
4. **Training:** Processes billions of examples and adjusts billions of parameters to predict the next word with increasing accuracy, gradually learning grammar, facts, and reasoning patterns.
5. **Doesn't think like a human:** When prompted, it does not reason or understand; it simply predicts the most likely next token based on learned patterns. Fine-tuning and reinforcement learning further refine its behavior. Ultimately, an LLM is a statistical engine that produces human-like text by modeling deep linguistic patterns, not by truly understanding the world.

Attention is all we need

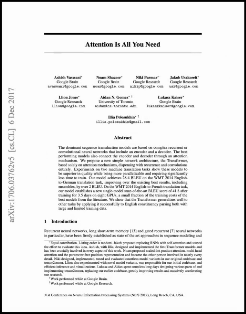


Attention is the core idea that lets a Transformer decide which parts of the input matter most to each other, regardless of distance.

Instead of reading a sequence step-by-step, the Transformer looks at all tokens at once and computes weighted relationships between them.

Generative Pre-trained Transformer (GPT)

LLMs do their job using GPT



arXiv:1706.03762v5 [cs.CL] 6 Dec 2017

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

1 Introduction

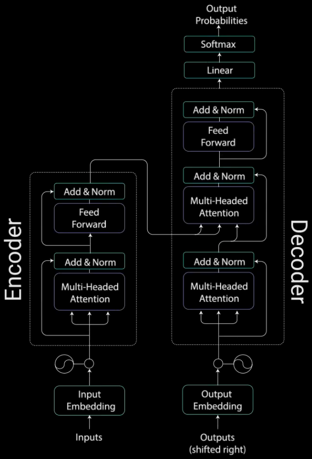
Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



Input Representation

Embedding and Positional Encoding

1. The model begins with a sequence of discrete input tokens representing the source data.
2. Each token is transformed into a fixed-length continuous vector using an embedding layer. These vectors capture semantic information in a high-dimensional space.
3. All token embeddings share the same dimensionality, known as the model dimension.
4. Because embeddings alone do not contain information about word order, an explicit positional encoding is generated for each token position.
5. The positional encoding assigns a unique pattern to each position, allowing the model to distinguish between tokens appearing at different locations in the sequence.
6. The final input to the encoder is obtained by adding the positional encoding to the corresponding token embeddings.
7. This combined representation contains both semantic meaning and positional information and is passed to the encoder stack.

Input Representation: Math Formulation

Embedding + Positional Encoding

$$X_{\text{emb}} = \text{Embed}(x_{1:N_{\text{src}}})$$

$$X_{\text{emb}} = \begin{bmatrix} \text{token } x_1 \\ \text{token } x_2 \\ \vdots \\ \text{token } x_{N_{\text{src}}} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathbf{e}_1 \in \mathbb{R}^{d_{\text{model}}} \\ \mathbf{e}_2 \in \mathbb{R}^{d_{\text{model}}} \\ \vdots \\ \mathbf{e}_{N_{\text{src}}} \in \mathbb{R}^{d_{\text{model}}} \end{bmatrix}$$

Each row of X_{emb} represents the embedding vector of a single input token.

The embedding dimension is the length of the vector used to represent each token (or item) in a continuous space and is typically $d_{\text{model}} \in \{512, 768, 1024, \dots\}$.

$$\boxed{P_{pos,\,2i} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad i \text{ even}}$$

$$\boxed{P_{pos,\,2i+1} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad i \text{ odd}}$$

$$P_{\text{src}} \in \mathbb{R}^{N_{\text{src}} \times d_{\text{model}}}$$

$$X = X_{\text{emb}} + P_{\text{src}} \quad \in \mathbb{R}^{N_{\text{src}} \times d_{\text{model}}}$$

Encoder

1. The encoder processes the entire input sequence simultaneously using self-attention.
2. For each token, the model computes multiple attention heads that determine how strongly the token should attend to every other token in the sequence.
3. Each attention head focuses on a different type of relationship, such as syntax, semantics, or long-range dependencies.
4. The outputs of all attention heads are combined to form a single representation for each token.
5. A residual connection and layer normalization are applied to stabilize training and preserve information from earlier layers.
6. A position-wise feedforward network is then applied independently to each token representation, introducing nonlinearity and additional expressive power.
7. A second residual connection and normalization produce the final encoder output.
8. The resulting representations encode the full source sequence and are passed to the decoder as contextual memory.

Encoder: Math Formulation

$$\boxed{X = X_{\text{emb}} + P_{\text{src}} \quad \in \mathbb{R}^{N_{\text{src}} \times d_{\text{model}}}}$$

$$\boxed{Q^{(i)} = X W_{Q,\text{enc}}^{(i)}, \quad K^{(i)} = X W_{K,\text{enc}}^{(i)}, \quad V^{(i)} = X W_{V,\text{enc}}^{(i)}}$$

$$\boxed{\text{Attn\_Head}_i = \text{softmax}\left(\frac{Q^{(i)}(K^{(i)})^\top}{\sqrt{d_k}}\right) V^{(i)}, \qquad i = 1, \ldots, h}$$

$$\boxed{O = \text{Concat}(\text{Attn\_Head}_1, \ldots, \text{Attn\_Head}_h)\, W_{O,\text{enc}}}$$

$$\mu = \frac{1}{d_{\text{model}}} \sum_{j=1}^{d_{\text{model}}} x_j, \quad \sigma = \sqrt{\frac{1}{d_{\text{model}}} \sum_{j=1}^{d_{\text{model}}} (x_j - \mu)^2 + \epsilon}, \quad \text{LN}(x) = \gamma \odot \frac{x - \mu}{\sigma} + \beta$$

$x \in \mathbb{R}^{d_{\text{model}}}$ (one token vector)

$$\boxed{X' = \text{LN}(X + O)}, \quad \boxed{F = \max(0, X' W_{1,\text{enc}} + b_{1,\text{enc}})\, W_{2,\text{enc}} + b_{2,\text{enc}}}, \quad \boxed{E = \text{LN}(X' + F)}$$

$E =$ Output of Encoder Block

Decoder

1. The decoder operates on the target sequence, which is generated one token at a time.
2. Target tokens are first embedded and combined with positional encodings, similar to the encoder input.
3. A masked self-attention mechanism ensures that each position can only attend to previously generated tokens, preventing access to future information.
4. The decoder then performs cross-attention, allowing each target position to attend to the encoded representations of the source sequence.
5. This cross-attention step aligns the target tokens with relevant parts of the input.
6. A feedforward network further refines each token representation.
7. Residual connections and layer normalization are applied after each major sublayer.
8. The final decoder output is projected into the vocabulary space to produce a probability distribution over the next possible token.
9. The token with the highest probability is selected or sampled to continue generation.

Decoder: Math Formulation

$$D_{\text{emb}} = \text{Embed}(y_{1:N_{\text{tgt}}}), \quad \boxed{D = D_{\text{emb}} + P_{\text{tgt}} \quad \in \mathbb{R}^{N_{\text{tgt}} \times d_{\text{model}}}}$$

$$\boxed{Q^{(i)} = DW^{(i)}_{Q,\text{dec}}, \quad K^{(i)} = DW^{(i)}_{K,\text{dec}}, \quad V^{(i)} = DW^{(i)}_{V,\text{dec}}}$$

$$\boxed{\text{Mask\_Attn\_Head}_i = \text{softmax}\left(\frac{Q^{(i)}(K^{(i)})^\top}{\sqrt{d_k}} + M\right) V^{(i)}, \ i = 1, \ldots, h \quad M_{ij} = \begin{cases} 0 & j \leq i \\ -\infty & j > i \end{cases}}$$

$$\boxed{O_1 = \text{Concat}(\text{Mask\_Attn\_Head}_1, \ldots, \text{Mask\_Attn\_Head}_h)\, W_{O,\text{dec}}}$$

$$\boxed{D' = \text{LN}(D + O_1)}$$

$$\boxed{Q^{(i)} = D'W^{(i)}_{Q,\text{cross}}, \quad K^{(i)} = EW^{(i)}_{K,\text{cross}}, \quad V^{(i)} = EW^{(i)}_{V,\text{cross}}}$$

$$\boxed{\text{Cross\_Attn\_Head}_i = \text{softmax}\left(\frac{Q^{(i)}(K^{(i)})^\top}{\sqrt{d_k}}\right) V^{(i)}}$$

$$\boxed{O_2 = \text{Concat}(\text{Cross\_Attn\_Head}_1, \ldots, \text{Cross\_Attn\_Head}_h)\, W_{O,\text{cross}}}$$

$$\boxed{D'' = \text{LN}(D' + O_2)}, \quad \boxed{F = \max(0, D''W_{1,\text{dec}} + b_{1,\text{dec}})\, W_{2,\text{dec}} + b_{2,\text{dec}}}, \quad \boxed{Y = \text{LN}(D'' + F)}$$

$$Y = \text{Output of Decoder Block}, \quad \boxed{Z = YW_{\text{vocab}} + b_{\text{vocab}},}, \quad \boxed{p(y_t) = \text{softmax}(Z_t)}$$

Why is Gen AI / LLM a revolution?

1. **New capability:** Machines can now generate text, code, images, insights, and decisions—beyond traditional automation.
2. **Natural language interface:** Business users can interact with systems using simple language, removing technical barriers.
3. **Reasoning at scale:** LLMs can analyse documents, summarise, compare, extract insights, and support decision-making.
4. **Productivity multiplier:** Dramatically accelerates work in research, coding, analytics, compliance, design, and communication.
5. **Foundation for autonomous workflows:** Enables AI agents that can plan, decide, and act with minimal human input.
6. **Democratisation of capability:** Advanced AI becomes accessible to every employee, not just specialists.

What can the LLMs do?

1. **Capabilities**
   - **Language Understanding & Generation:** Conversation, Q&A, summarisation, translation, rewriting, content creation.
   - **Reasoning & Problem-Solving:** Logical reasoning, planning, multi-step tasks, explanations.
   - **Knowledge & Information:** Concept explanation, knowledge recall, information extraction.
   - **Coding & Technical Tasks:** Code generation, debugging, optimisation, documentation.
   - **Data & Document Intelligence:** Table interpretation, document classification, entity extraction.
   - **Multimodal Capabilities:** Image understanding, speech processing, image/audio generation (model-dependent).
   - **RAG (Retrieval-Augmented Generation):** Search, grounding answers, reducing hallucinations.
   - **Agentic Behaviour:** Tool use, workflow automation, copilots.
   - **Personalisation & Recommendations:** User-specific responses, adaptive learning.
2. **Limitations**: Not infallible
   - May produce incorrect or fabricated information (“hallucinations”)
   - Outputs are based on learned patterns, not real understanding
   - No built-in fact checking or awareness of truth
3. **Bottom Line**: Very powerful, but must be used with human verification.

Some Large Language Models

Overview

1. Foundation models: Gemini, GPT, Claude, Llama
2. Multimodal AI: text, image, audio, video
3. Agentic systems: planning, reasoning, tool use

Model Lineage & Architecture

1. **GPT (OpenAI)** – Dense/hybrid Transformer; natively multimodal (text–image–audio); strong reasoning & tool integration.
2. **Gemini (Google)** – Fully multimodal from inception; strong video/audio understanding; hierarchical attention design.
3. **Claude (Anthropic)** – Sparse/MoE-enhanced Transformer; long-context specialist; trained with Constitutional AI.
4. **Llama (Meta)** – Efficient decoder-only Transformer; open-source; optimized for local, on-device & edge inference.

Key Technical Distinctions

1. **Multimodality:** GPT & Gemini are native; Claude uses encoders + fusion; Llama relies on adapters.
2. **Training Philosophy:** GPT uses synthetic data + tool traces; Claude uses Constitutional AI; Gemini leverages large multimodal corpora; Llama emphasizes openness & efficiency.
3. **Context Windows:** Claude & Gemini lead (1M+ tokens); GPT supports 200k; Llama smaller but extendable.

Closing Thoughts & Questions

AI is not just tools or code,
but how we choose each path as we go.
It shifts how work and worth align,
and redraws value over time.

Thank you!