Shakil Mahmud • Research Monograph

SHAKIL MAHMUD EDITORIAL & RESEARCH MONOGRAPHS

Large Language Models & Generative AI

From subword tokenization, scaling laws, and distributed Megatron/ZeRO clusters to FlashAttention, LoRA, RLHF, DPO, diffusion ODEs, and multimodal foundation systems.

SHAKIL MAHMUD

Independent Scholar & Systems Researcher

SHAKIL MAHMUD EDITORIAL & RESEARCH SERIES

First Collected Edition • Dhaka & Singapore • 2026

Large Language Models & Generative AI

From subword tokenization, scaling laws, and distributed Megatron/ZeRO clusters to FlashAttention, LoRA, RLHF, DPO, diffusion ODEs, and multimodal foundation systems.

LIBRARY & CATALOGING-IN-PUBLICATION DATA

Title: Large Language Models & Generative AI
Author: Shakil Mahmud
Series: Shakil Mahmud Editorial & Research Series (Monograph Collection)
Subjects: Science -- History -- Philosophy -- Mathematics -- Civilization
Published: Dhaka & Singapore, 2026
Format: Digital Monograph Offprint (PDF) • Complete 20-Part Collected Edition
Permanent URI: https://shakilmahmud.com/series/generative-ai/

Suggested Academic Citation:
Mahmud, Shakil. (2026). *Large Language Models & Generative AI: The Complete 20-Part Collected Monograph*. Shakil Mahmud Research Series. https://shakilmahmud.com/series/generative-ai/

Open Scholarly Access: This collected monograph is published under Open Scholarly Access for personal study, academic citation, university pedagogy, and non-commercial research.

© 2026 Shakil Mahmud. All rights reserved. Typeset and published at Dhaka and Singapore.

Shakil Mahmud • Research Monograph

EDITORIAL PREFACE

Preface: The Scaffolding of Reality

Generative AI and Large Language Models represent the computational synthesis of human symbolic cognition. From subword Byte-Pair tokenization, emergent neural scaling laws, FlashAttention tiling, and sparse mixture-of-experts to low-rank parameter-efficient adaptation, reinforcement learning from human feedback, direct preference optimization, and stochastic continuous-time diffusion, this monograph delivers the definitive technical blueprint of foundation models.

The purpose of this collected monograph is not merely didactic exposition, but the systematic synthesis of conceptual revolutions. Each chapter examines an intellectual threshold where old paradigms faltered and new mathematical or physical insight illuminated deeper structures of reality.

— Shakil Mahmud
Dhaka & Singapore, 2026

INDEX OF PARTS

Table of Contents

PHASE I: TOKENIZATION & COMPUTE SCALING LAWS

PART I — The Tokenization Bottleneck — Byte-Pair Encoding, WordPiece & Subwords — 563 words
The Tokenization Bottleneck — Byte-Pair Encoding, WordPiece & Subwords

PART II — Scaling Laws of Neural Language Models — Kaplan vs Chinchilla Compute Optimality — 604 words
Scaling Laws of Neural Language Models — Kaplan vs Chinchilla Compute Optimality

PART III — Dense Transformer Architectures — GPT-3, PaLM, LLaMA & The Modern Recipe — 555 words
Dense Transformer Architectures — GPT-3, PaLM, LLaMA & The Modern Recipe

PART IV — The KV Cache & Inference Latency — PagedAttention, vLLM & Serving Memory — 587 words
The KV Cache & Inference Latency — PagedAttention, vLLM & Serving Memory

PHASE II: FLASH-ATTENTION & SYSTEM ARCHITECTURE

PART V — Multi-Query & Grouped-Query Attention — Breaking the Memory Bandwidth Bound — 549 words
Multi-Query & Grouped-Query Attention — Breaking the Memory Bandwidth Bound

PART VI — FlashAttention & IO-Awareness — GPU SRAM Tiling & Mathematical Online Softmax — 554 words
FlashAttention & IO-Awareness — GPU SRAM Tiling & Mathematical Online Softmax

PART VII — Model Quantization Theory — FP16, INT8, INT4, GPTQ, AWQ & Outlier Dimensions — 443 words
Model Quantization Theory — FP16, INT8, INT4, GPTQ, AWQ & Outlier Dimensions

PART VIII — Parameter-Efficient Fine-Tuning — LoRA, QLoRA & The Geometry of Intrinsic Rank — 470 words
Parameter-Efficient Fine-Tuning — LoRA, QLoRA & The Geometry of Intrinsic Rank

PHASE III: SPARSE MOE & PARAMETER-EFFICIENT ADAPTATION

PART IX — Sparse Mixture-of-Experts (MoE) — Router Gates, Load Balancing & Switch Transformers — 587 words
Sparse Mixture-of-Experts (MoE) — Router Gates, Load Balancing & Switch Transformers

PART X — Pretraining Dynamics & Data Engineering — MinHash Deduplication & Synthetic Curations — 434 words
Pretraining Dynamics & Data Engineering — MinHash Deduplication & Synthetic Curations

PART XI — Supervised Fine-Tuning & Multi-Task Instruction Tuning — Cross-Entropy Dynamics — 334 words
Supervised Fine-Tuning & Multi-Task Instruction Tuning — Cross-Entropy Dynamics

PART XII — Reinforcement Learning from Human Feedback (RLHF) — Bradley-Terry Reward Modeling & PPO — 451 words
Reinforcement Learning from Human Feedback (RLHF) — Bradley-Terry Reward Modeling & PPO

PART XIII — Direct Preference Optimization (DPO) — Bypassing the Reward Model with Implicit Log-Odds — 512 words
Direct Preference Optimization (DPO) — Bypassing the Reward Model with Implicit Log-Odds

PART XIV — Prompt Engineering & In-Context Learning — Induction Heads, Chain-of-Thought & Emergence — 432 words
Prompt Engineering & In-Context Learning — Induction Heads, Chain-of-Thought & Emergence

PHASE IV: ALIGNMENT, DIFFUSION & FRONTIER SYSTEMS

PART XV — Retrieval-Augmented Generation (RAG) — Dense Embeddings, HNSW Indexing & Hybrid Reranking — 401 words
Retrieval-Augmented Generation (RAG) — Dense Embeddings, HNSW Indexing & Hybrid Reranking

Shakil Mahmud • Research Monograph

PART XVI	**Autoregressive Decoding Strategies — Temperature, Top-P Nucleus & Speculative Decoding** *Autoregressive Decoding Strategies — Temperature, Top-P Nucleus & Speculative Decoding*	454 words
PART XVII	**Score-Based Continuous Generative Diffusion — Stochastic Differential Equations & DDPM** *Score-Based Continuous Generative Diffusion — Stochastic Differential Equations & DDPM*	434 words
PART XVIII	**Latent Diffusion & Flow Matching — Stable Diffusion, Perceptual Compression & Rectified Flows** *Latent Diffusion & Flow Matching — Stable Diffusion, Perceptual Compression & Rectified Flows*	448 words
PART XIX	**Long-Context Systems — RingAttention, Blockwise Context & Needle-In-A-Haystack Limits** *Long-Context Systems — RingAttention, Blockwise Context & Needle-In-A-Haystack Limits*	505 words
PART XX	**The Frontiers of Synthetic Cognition — Multimodality, Reasoning Tokens & Test-Time Compute** *The Frontiers of Synthetic Cognition — Multimodality, Reasoning Tokens & Test-Time Compute*	404 words

SHAKIL MAHMUD RESEARCH MONOGRAPHS · PART I

The Tokenization Bottleneck — Byte-Pair Encoding, WordPiece & Subwords

The Tokenization Bottleneck — Byte-Pair Encoding, WordPiece & Subwords

563 words • November 4, 2026 • Permanent URI

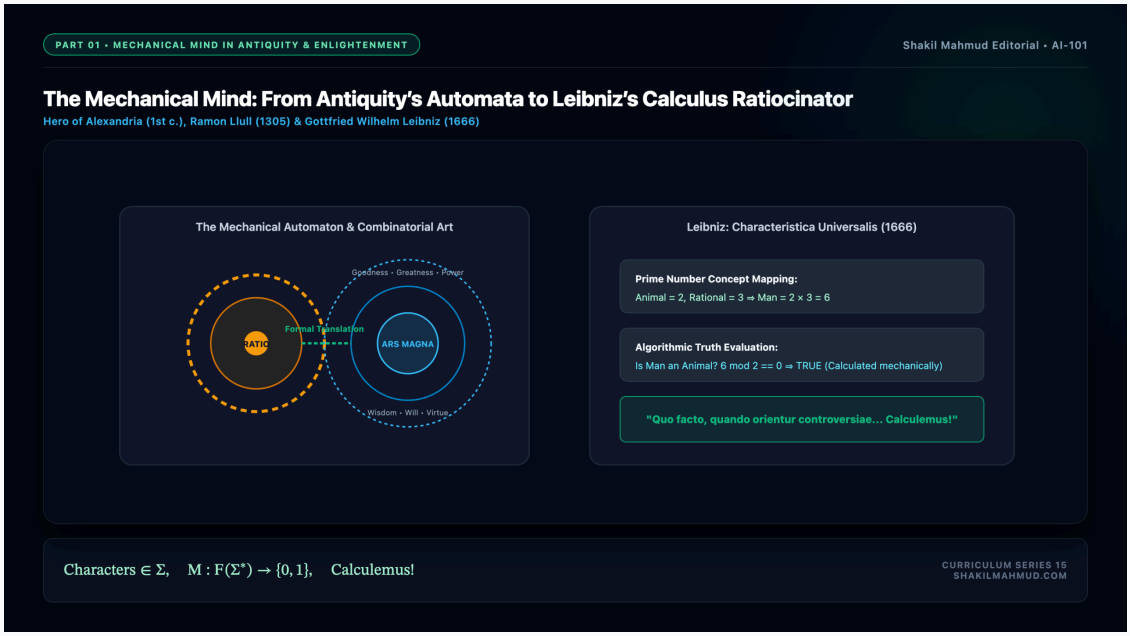


Plate I: *Mechanical automata and Ramon Llull's Ars Magna combinatorial rings alongside Gottfried Wilhelm Leibniz's Characteristica Universalis and Calculus Ratiocinator.*

Before a modern Large Language Model (LLM) performs its first floating-point calculation, continuous human language must undergo a discrete symbolic quantization: **Tokenization**. Far from being a trivial preprocessing step, tokenization represents the foundational inductive bottleneck of generative AI. Pathologies commonly blamed on neural attention—such as arithmetic failures, character-level blindness, regex hallucinations, and multilingual cost disparities—originate almost entirely in the discrete subword vocabulary algorithms that slice text into token IDs.

1. The Fundamental Dilemma: Characters vs Words

In natural language processing, representing symbolic sequences confronts an unavoidable trade-off:

- **Word-Level Tokenization:** Assigning an integer to every unique word yields short sequence lengths N, but induces an astronomical, open-ended vocabulary size (|mathcal{V}| > 10^7). Morphologically rich languages (Finnish, Turkish, Bengali) produce millions of unseen out-of-vocabulary (OOV) tokens, which collapse into the catch-all unknown token $langletextunkrangle$.
- **Character-Level Tokenization:** Maintaining a tiny vocabulary ($|mathcalV|approx256$) eliminates OOV errors, but inflates sequence length N by a factor of 4 to 10. Because Transformer attention scales quadratically ($O(N^2)$), character-level modeling imposes crushing computational overhead and dilutes semantic attention patterns across hundreds of low-information tokens.

2. Byte-Pair Encoding (BPE): Algorithmic Formulation

In 2016, Rico Sennrich, Barry Haddow, and Alexandra Birch adapted Philip Gage's 1994 data compression algorithm, **Byte-Pair Encoding (BPE)**, to solve open-vocabulary neural machine translation.

Given a raw training corpus $mathcalD$, BPE operates iteratively:

Shakil Mahmud • Research Monograph

1. Initialize vocabulary $mathcalV_0$ as the set of all unique atomic characters in $mathcalD$. Represent every word as a sequence of space-delimited characters followed by a special end-of-word marker $langletextwrangle$.
2. At each iteration step $t = 1, dots, K$:
   Identify the most frequent contiguous adjacent symbol pair (c_i, c_j) across the entire corpus:

   $$(c_1^*, c_2^*) = argmax_{(c_i,c_j)inmathcalV_timesmathcalV_t} textCount(c_i, c_j; mathcalD)$$

3. Merge the winning pair into a single atomic subword token: $c_textnew = c_1^* circ c_2^*$.
4. Expand vocabulary: $mathcalV_t+1 = mathcalV_t cup c_textnew$, and replace all occurrences of (c_1^*, c_2^*) in $mathcalD$ with $c_textnew$.

Frequent words (e.g., “the”, “transformation”) merge into single monolithic tokens, while rare or novel words (e.g., “antigravity”, “decentralized”) decompose into meaningful morphological subwords: `['anti', 'gravity']`, preserving open-vocabulary coverage.

3. Byte-Level BPE (BBPE) and Universal Fallbacks

In GPT-2 (2019) and modern successors (LLaMA, Mistral, GPT-4), OpenAI generalized BPE to the raw byte level: **Byte-Level BPE**. The base alphabet is initialized strictly with the 256 unique values of an 8-bit byte ($0x00 dots 0xFF$).

Because any UTF-8 string is fundamentally a sequence of bytes, BBPE guarantees that **no string in any human language or binary encoding can ever produce an out-of-vocabulary error**. Furthermore, regex-based pre-tokenization rules prevent merges across punctuation boundaries and whitespace, preventing cross-domain syntax contamination.

4. The Architectural Consequences of Tokenization

Despite its elegance, subword tokenization imposes subtle computational pathologies:

- **Arithmetic Degradation:** Numbers are tokenized arbitrarily based on corpus frequency. For example, `12345` might tokenize as `['12', '345']` while `67890` becomes `['678', '90']`. The neural network cannot easily align decimal places, causing basic addition to fail.
- **Spelling Blindness:** Because the token `strawberry` is ingested as an atomic embedding vector e_4512, the model never accesses its individual letters, making questions like *“How many ‘r’s are in strawberry?”* deceptively difficult.
- **The Multilingual Token Tax:** English text compresses efficiently at $sim1.3$ tokens per word, whereas Bengali, Hindi, Arabic, and Japanese frequently require 4 – 8 tokens per word due to underrepresentation in the BPE merge hierarchy, effectively quadrupling API inference costs and halving context window capacity for non-English speakers.

Citation: Mahmud, S. (2026). "The Tokenization Bottleneck — Byte-Pair Encoding, WordPiece & Subwords." In Large Language Models & Generative AI, Part 1.
https://shakilmahmud.com/large-language-models-part-01-tokenization-byte-pair-encoding-bpe-subwords/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART II

Scaling Laws of Neural Language Models — Kaplan vs Chinchilla Compute Optimality

Scaling Laws of Neural Language Models — Kaplan vs Chinchilla Compute Optimality

604 words • November 4, 2026 • Permanent URI

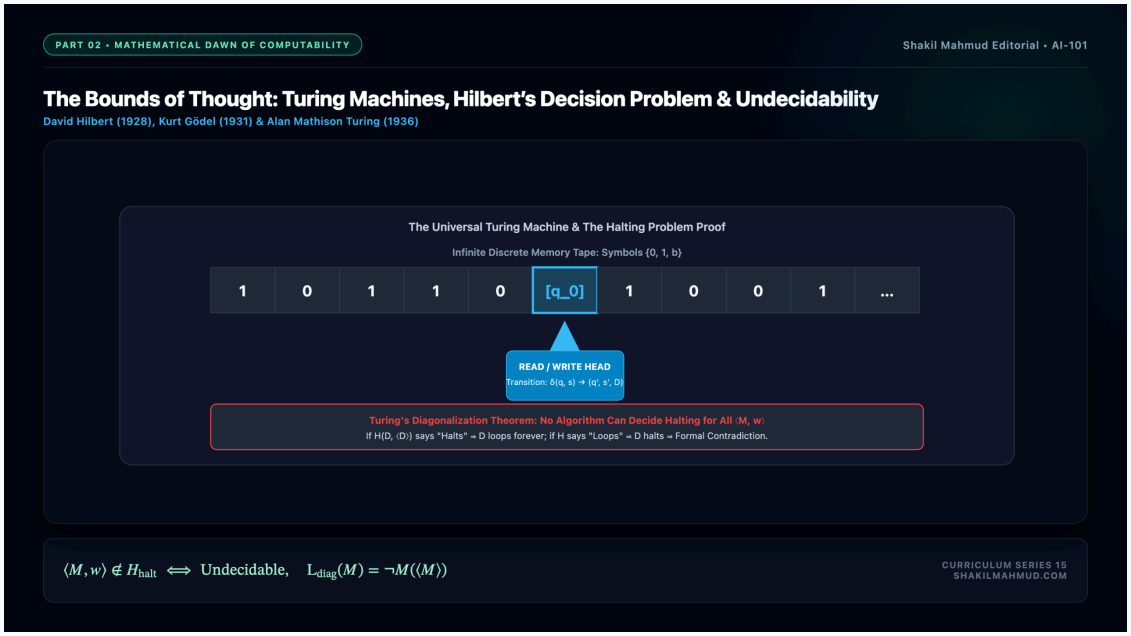


Plate II: *The Universal Turing Machine discrete tape architecture and the diagonal argument establishing the undecidability of the Halting Problem.*

Throughout the history of computer science, algorithmic performance was predominantly governed by engineering cleverness and handcrafted feature representations. Large Language Models shattered this assumption through the discovery of **Neural Scaling Laws**: cross-entropy loss does not fluctuate unpredictably; it decays as a mathematically pristine, scale-invariant power-law function of compute budget, dataset size, and parameter count across eight orders of magnitude. The race for foundation models is fundamentally governed by the thermodynamics of compute optimality.

1. The Kaplan Scaling Laws (OpenAI, 2020)

In 2020, Jared Kaplan et al. at OpenAI published the first comprehensive empirical study of Transformer scaling dynamics. When performance is not bottlenecked by dataset size D or parameter count N, test cross-entropy loss L adheres strictly to continuous power-law relations:

$$L(N) = left(fracN_cNright)^alpha_N, quadL(D) = left(fracD_cDright)^alpha_D, quadL(C) = left(fracC_cCright)^alpha_C$$

where $Capprox6ND$ estimates total floating-point operations (FLOPs) required to train a dense model with N non-embedding parameters across D tokens ($2textFLOPs/param$ on the forward pass, $4textFLOPs/param$ on the backward pass).

Kaplan et al. fitted the empirical exponents as $alpha_Napprox0.076$, $alpha_Dapprox0.096$, and concluded that models should scale parameters far more aggressively than training tokens:

$$N_textoptproptoC^0.73, quadD_textoptproptoC^0.27$$

Shakil Mahmud • Research Monograph

This empirical law led to the design of first-generation massive foundation models: GPT-3 (175 billion parameters trained on only 300 billion tokens), Gopher (280B on 300B tokens), and MT-NLG (530B on 270B tokens).

2. The Chinchilla Paradigm Shift (DeepMind, 2022)

In 2022, Jordan Hoffmann et al. at Google DeepMind revisited Kaplan's scaling formulations, identifying a fatal methodological flaw: Kaplan had maintained a fixed learning rate cosine decay schedule while varying compute budgets, severely disadvantaging larger token runs.

Hoffmann et al. fitted an explicit parametric loss envelope over 400 experimental training runs spanning 10^{18} to 10^{24} FLOPs:

$$L(N, D) = E + fracAN^alpha + fracBD^beta$$

where $E = 1.69$ is the irreducible entropy of the English language, $A = 406.4$, $B = 410.7$, $alpha = 0.34$, and $beta = 0.28$.

Minimizing $L(N, D)$ subject to total compute constraint $C = 6ND$ via the method of Lagrange multipliers:

mathcal{L}(N, D, lambda) = E + A N^{-alpha} + B D^{-beta} + lambda (6 N D – C)

Evaluating partial derivatives $fracpartialmathcalLpartialN = -alphaAN^-alpha-1 + 6lambdaD = 0$ and $fracpartialmathcalLpartialD = -betaBD^-beta-1 + 6lambdaN = 0$ yields the profound **Chinchilla Scaling Law**:

$$N_textopt proptoC^fracbetaalpha+beta approxC^0.5, quad D_textopt proptoC^fracalphaalpha+beta approxC^0.5$$

3. The Chinchilla Rule of Thumb: 1:1 Parity

Because $alphaapproxbetaapprox0.3$, parameters and tokens should scale in exact **1:1 parity**:

$$fracD_textoptN_textoptapproxfracGalpha/betaapprox20$$

For compute-optimal pretraining, **a model must be trained on approximately 20 tokens per parameter**.

DeepMind demonstrated this by training **Chinchilla** (70 billion parameters on 1.4 trillion tokens). Despite being less than half the size of GPT-3 (175B), Chinchilla crushed GPT-3, Gopher, and MT-NLG across every downstream linguistic, mathematical, and reasoning benchmark while consuming half the inference memory.

4. Inference-Optimal Over-Training: LLaMA and Modern Scale

Chinchilla compute-optimality minimizes total FLOPs consumed during *pretraining*. In real-world enterprise deployments, however, models serve billions of inference requests over months. A smaller model trained far beyond the Chinchilla frontier achieves lower per-token serving latency and lower HBM memory footprints.

In 2023, Meta introduced **LLaMA** (Touvron et al.), pioneering *inference-optimal over-training*: training a 7B model on 1.0T tokens ($140texttokens/param$) and a 70B model on 2.0T tokens ($30texttokens/param$), proving that loss continues to decline smoothly far past Chinchilla boundaries without saturation.

Citation: Mahmud, S. (2026). "Scaling Laws of Neural Language Models — Kaplan vs Chinchilla Compute Optimality." In Large Language Models & Generative AI, Part 2.
https://shakilmahmud.com/large-language-models-part-02-neural-scaling-laws-kaplan-chinchilla-optimality/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART III

Dense Transformer Architectures — GPT-3, PaLM, LLaMA & The Modern Recipe

Dense Transformer Architectures — GPT-3, PaLM, LLaMA & The Modern Recipe

555 words • November 4, 2026 • Permanent URI

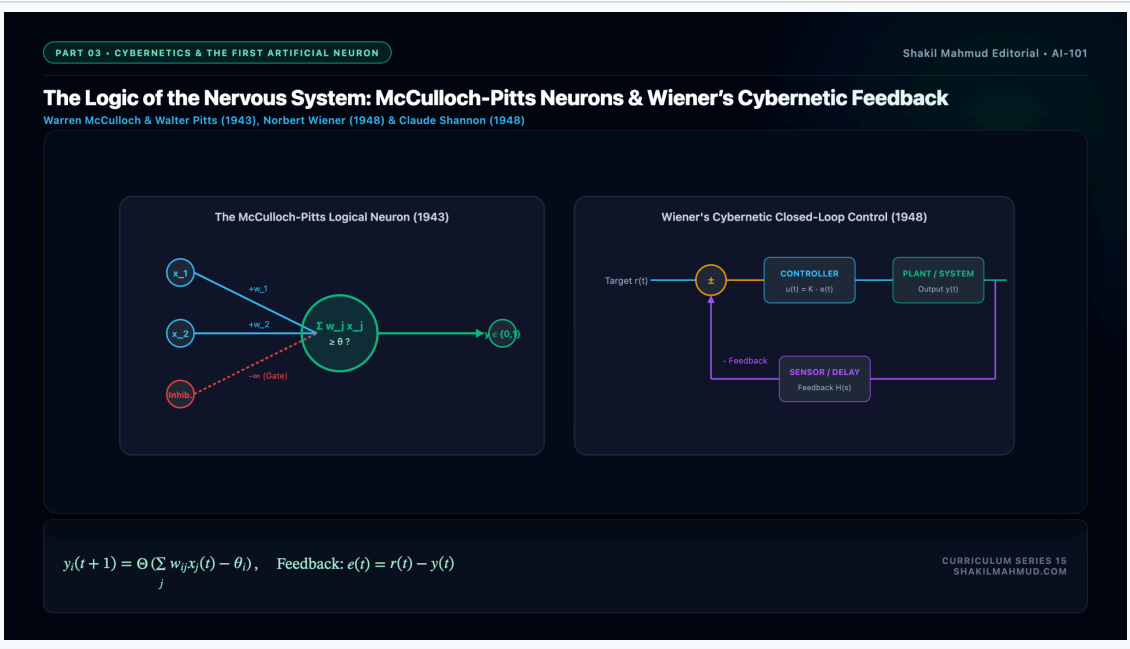


Plate III: *McCulloch-Pitts binary threshold neuron computing propositional logic, paired with Norbert Wiener's cybernetic negative feedback control loop.*

Between the publication of the original Transformer in 2017 and the release of LLaMA and Mistral in 2023, the foundational architectural blueprint of deep autoregressive language models underwent a silent, systematic Darwinian evolution. While early models (GPT-1 through GPT-3) adhered closely to Vaswani's initial formulations, modern trillion-parameter foundation models have converged onto an ultra-optimized, unified architectural standard: **Pre-RMSNorm, SwiGLU non-linearities, Rotary Position Embeddings (RoPE), and untied vocabulary projections**.

1. The Evolution of Autoregressive Decoder Blocks

A causal autoregressive language model decomposes the joint probability of a sequence $X = (x_1, dots, x_T)$ via the chain rule of probability:

$$P(X) = prod_{t=1}^{T} P(x_t mid x_1, dots, x_{t-1})$$

To enforce causality, self-attention applies a lower-triangular causal attention mask $Minmathbb{R}^{TtimesT}$ where $M_{ij} = -infty$ for j > i, preventing queries at position i from accessing future keys j > i.

Over six years of rigorous industrial ablation, four architectural choices were definitively discarded:

1. **Post-LN to Pre-LN:** Moving LayerNorm from the outer residual sum directly into the sublayer input branches eliminates gradient vanishing across depth.
2. **LayerNorm to RMSNorm:** Removing mean-centering accelerates forward and backward passes by 10 without degradation.
3. **ReLU/GELU to SwiGLU:** Replacing standard feed-forwards with gated bilinear projections substantially increases parameter expressivity.

Shakil Mahmud • Research Monograph

4. **Absolute Positional Encodings to RoPE:** Replacing static positional vectors with multiplicative rotary embeddings unlocks context extrapolation.

2. SwiGLU: Gated Linear Units and Expressive Volume

In 2020, Noam Shazeer at Google introduced **GLU (Gated Linear Unit) Variants** for Transformers. In standard architectures, the FFN applies a single non-linear projection: $textFFN(x) = textGELU(xW_1 + b_1)W_2 + b_2$.

Shazeer's **SwiGLU (Swish Gated Linear Unit)** splits the input through two parallel linear projections, gating one with the continuous Swish/SiLU activation function $textswish(z) = zcdotsigma(betaz)$ before element-wise multiplication:

$$textSwiGLU(x) = left(textswish(xW_1)odot(xW_2)right)W_3$$

To maintain identical parameter parity with standard $4d_{textmodel}$ expansion, SwiGLU sets the intermediate hidden dimension to $d_{textffn} = frac83d_{textmodel}$. Empirically, SwiGLU delivers lower perplexity at identical compute budgets, becoming the universal standard in PaLM, LLaMA, Mistral, and Gemma.

3. Parameter Allocation and Layer Sizing

Given total parameter budget N, how should layers, hidden width, and attention heads be sized? Let L be depth, d be hidden dimension, and V be vocabulary size. Non-embedding parameters are dominated by:

$$N_{textnon-emb}approx12Ld^2$$

Across the historical frontier, empirical balance favored wide, deep networks with key dimension $d_k = d/h = 128$:

- **GPT-3 (175B):** $L = 96, d = 12288, h = 96, d_k = 128$
- **PaLM (540B):** $L = 118, d = 18432, h = 48, d_k = 256$
- **LLaMA-3 (70B):** $L = 80, d = 8192, h = 64, d_k = 128, V = 128256$

4. Untied Output Embeddings: Discarding Weight Tying

In early language models (Press & Wolf, 2017), the input token embedding matrix $W_einmathbbR^Vtimesd$ was shared with the final unembedding projection head: $W_{texthead} = W_e^T$. While weight tying saves $Vtimesd$ parameters when $|V|$ is small, modern models employ massive token vocabularies ($V = 128textk$ to $256textk$).

Chung et al. (2020) demonstrated that tying input and output representations forces an artificial symmetry: input embeddings seek semantic clustering, whereas output logits require sharp geometric separation to assign near-zero probabilities to irrelevant words. Decoupling (untying) W_e and $W_{texthead}$ improves downstream reasoning accuracy across scale.

Citation: Mahmud, S. (2026). "Dense Transformer Architectures — GPT-3, PaLM, LLaMA & The Modern Recipe." In Large Language Models & Generative AI, Part 3.
https://shakilmahmud.com/large-language-models-part-03-dense-transformer-architectures-llama-mistral-paradigms/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART IV

The KV Cache & Inference Latency — PagedAttention, vLLM & Serving Memory

The KV Cache & Inference Latency — PagedAttention, vLLM & Serving Memory

587 words • November 4, 2026 • Permanent URI

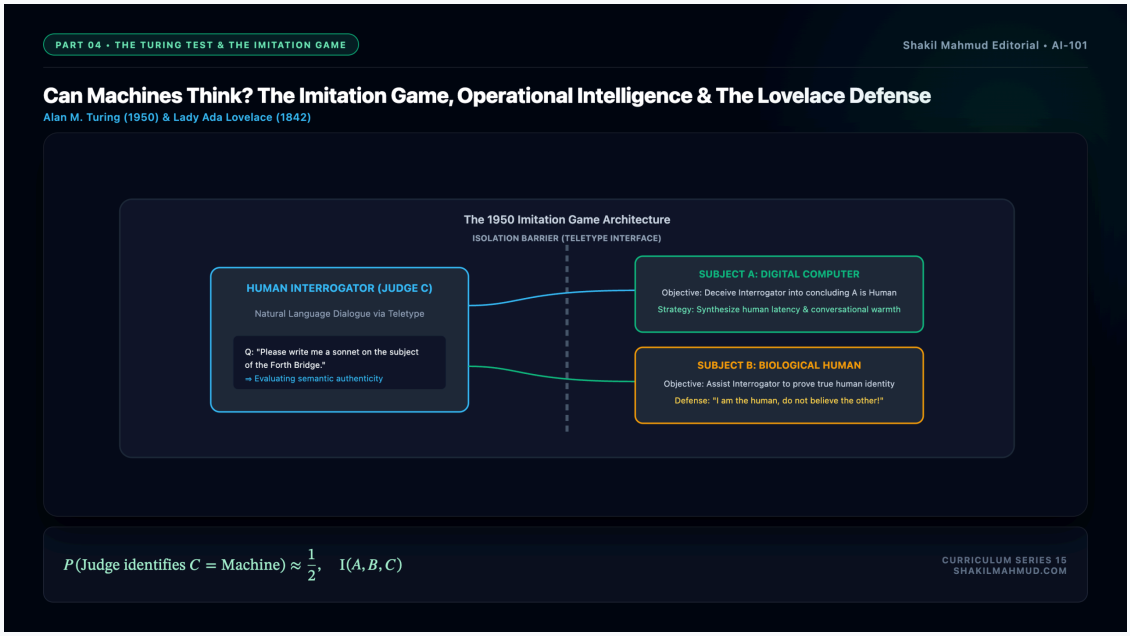


Plate IV: *Alan Turing's 1950 Imitation Game tripartite architecture, evaluating machine intelligence through blind teletype interrogation.*

Training a large language model is an overwhelmingly compute-bound process dominated by massive matrix multiplications running on parallel Tensor Cores. Serving that same model during autoregressive inference, however, undergoes a complete thermodynamic inversion: token generation is severely **Memory-Bandwidth Bound**. For every single generated token, the model must read all billions of static model parameters plus gigabytes of dynamic history from GPU memory. The primary battleground of modern inference systems is the **Key-Value (KV) Cache** and memory management architectures like **PagedAttention (vLLM)**.

1. The Autoregressive Generation Bottleneck: Prefill vs Decode

Inference proceeds in two fundamentally distinct operational regimes:

1. **The Prefill (Prompt) Phase:** Ingests prompt tokens $(x_1, dots, x_P)$ simultaneously in parallel. The entire $PtimesP$ causal attention matrix is evaluated in a single forward pass, completely saturating GPU arithmetic units (Compute-Bound, High Arithmetic Intensity).
2. **The Decode (Generation) Phase:** Generates tokens autoregressively one by one $(x_{P+1}, x_{P+2}, dots)$. At step t, the query vector is a single token vector $q_tinmathbbR^{1timesd}$. Evaluating attention against past context requires accessing keys and values for all preceding positions $1dotst-1$.

Without caching, past hidden states would have to be recomputed from scratch at every step, scaling generation cost as $O(T^2)$. Storing past keys and values in GPU VRAM avoids recomputation: this is the **KV Cache**.

2. Anatomy of the KV Cache Footprint

For a model with $n_{textlayers}$ layers, $n_{textheads}$ key/value heads, and head dimension $d_{texthead}$, storing 16-bit floating-point (FP16/BF16, 2 bytes) representations across batch size b and sequence length s requires:

Shakil Mahmud • Research Monograph

$$textMemory_{textKV} = 2times2timesn_{textlayers}timesn_{textheads}timesd_{texthead}timesbtimessquadtextBytes$$

For a standard LLaMA-2 70B model ($n_{textlayers} = 80, n_{textheads} = 64, d_{texthead} = 128$) running batch size $b = 32$ with context length $s = 4096$:

$$textMemory_{textKV} = 4times80times64times128times32times4096 = 343.6times10^9textBytesapprox344textGB$$

The KV cache alone exceeds the memory capacity of four 80GB NVIDIA A100 GPUs, surpassing the static parameter weights ($140textGB$) of the model itself!

3. Memory Fragmentation in Traditional Serving

Prior to 2023, serving frameworks (HuggingFace TGI, FasterTransformer) allocated contiguous chunks of virtual memory based on the *maximum possible request length* (e.g., reserving 2048 tokens). Because actual user queries are unpredictable in length, this design caused catastrophic memory fragmentation:

- **Internal Fragmentation:** Unused pre-allocated slots reserved for tokens that were never generated.
- **External Fragmentation:** Scattered non-contiguous memory chunks too small to accommodate new requests.

Empirical audits revealed that **60 to 80 of GPU VRAM was wasted** on empty reserved memory, forcing batch sizes down and crippling throughput.

4. PagedAttention and vLLM: Virtual Memory for LLMs

In 2023, Woosuk Kwon et al. at UC Berkeley resolved this crisis with **PagedAttention**, drawing inspiration from classical operating system virtual memory and paging.

PagedAttention partitions the KV cache into fixed-size physical memory blocks (typically 16 or 32 tokens). Instead of storing tokens contiguously, blocks are allocated dynamically on-demand from a centralized block table: logical token positions map to arbitrary physical GPU memory locations.

Key algorithmic innovations of PagedAttention include:

- **Near-Zero Memory Waste:** Memory waste drops below 4, occurring only in the final block of a sequence. GPU serving concurrency and batch size increase by $2times$ to $4times$.
- **Zero-Copy Branching:** Parallel sampling (e.g., beam search, multiple candidate completions) shares physical blocks using *Copy-on-Write (CoW)*, consuming zero additional memory until branches diverge.
- **Prefix Caching:** Frequently recurring system prompts, multi-turn chat histories, or document contexts reside persistently in cache blocks, eliminating redundant prefill computation entirely.

Citation: Mahmud, S. (2026). "The KV Cache & Inference Latency — PagedAttention, vLLM & Serving Memory." In Large Language Models & Generative AI, Part 4.
https://shakilmahmud.com/large-language-models-part-04-kv-cache-inference-latency-pagedattention-vllm/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART V

Multi-Query & Grouped-Query Attention — Breaking the Memory Bandwidth Bound

Multi-Query & Grouped-Query Attention — Breaking the Memory Bandwidth Bound

549 words • November 4, 2026 • Permanent URI

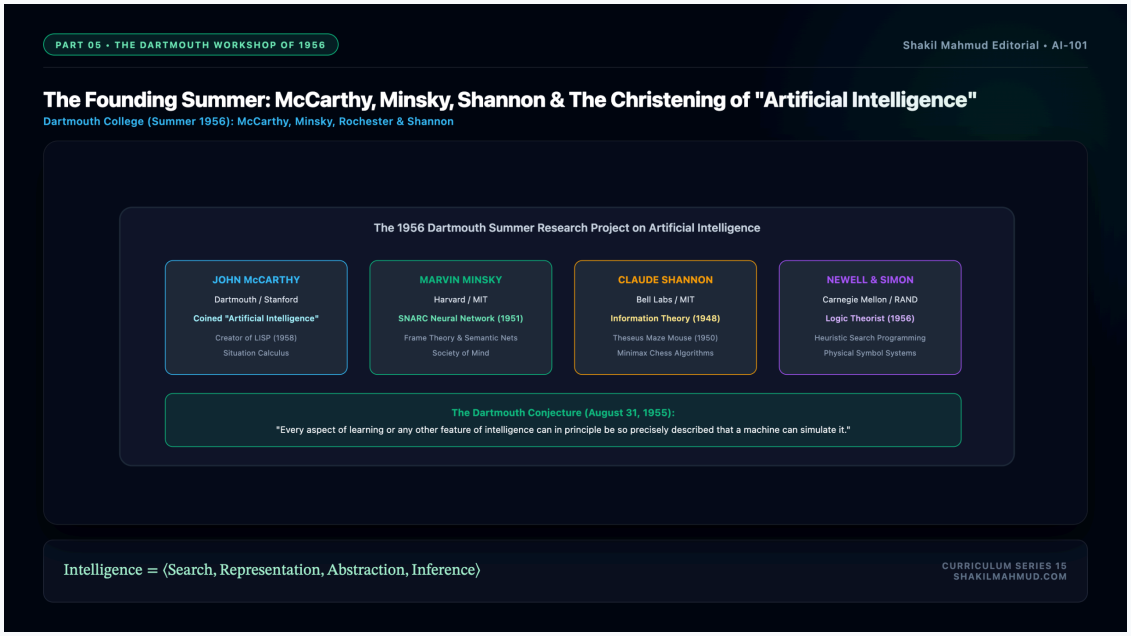


Plate V: *The 1956 Dartmouth Summer Research Project on Artificial Intelligence: McCarthy, Minsky, Shannon, and Newell-Simon founding the discipline.*

During autoregressive token generation, Large Language Models are rarely bounded by raw floating-point compute; they are severely throttled by **Memory Bandwidth**. For every single generated token, the model must read all its weights and the entire growing history of Key-Value tensors from High Bandwidth Memory (HBM) into on-chip cache. **Multi-Query Attention (MQA)** and **Grouped-Query Attention (GQA)** represent revolutionary structural innovations designed to slash the KV-cache bandwidth footprint by up to 8x without sacrificing representational capacity.

1. The Autoregressive Memory Wall & Arithmetic Intensity

The operational efficiency of neural inference is governed by **Arithmetic Intensity**: the ratio of floating-point operations (FLOPs) performed to bytes transferred from memory:

$$mathcalI = fractextFLOPstextBytesTransferred$$

In standard Multi-Head Attention (MHA), given hidden dimension d, number of query heads H_Q, and sequence length S, each head maintains its own Key and Value projections:

$$mathbfK_h = mathbfXmathbfW_K^(h), quadmathbfV_h = mathbfXmathbfW_V^(h)quad(h = 1, dots, H_Q)$$

During decoding of a single token ($N = 1$), computing the attention output requires fetching the entire historical KV-cache of size $2timesH_QtimesStimesd_k$ bytes. Because the FLOPs scale as $O(S)$ while memory traffic also scales as $O(S)$, the arithmetic intensity drops below 1 FLOP/byte. Modern accelerators (NVIDIA H100) boast 2,000 TFLOPs of FP16 compute but only 3.35 TB/s of memory bandwidth. Under MHA decoding, tensor cores sit idle more than 90% of the time, starved for data.

Shakil Mahmud • Research Monograph

2. Multi-Query Attention (MQA): Extreme KV Sharing

Introduced by Noam Shazeer in 2019, **Multi-Query Attention (MQA)** eliminates head-specific Key and Value projections entirely. While Query heads remain independent (H_Q heads), all Query heads share a single common Key head and Value head ($H_{KV} = 1$):

$mathbfQinmathbbR^{BtimesH_Qtimes1timesd_k}, quadmathbfKinmathbbR^{Btimes1timesStimesd_k}, quadmathbfVinmathbbR^{Btimes1timesStimesd_k}$

The attention computation broadcasts the single Key and Value tensor across all H_Q query heads:

$mathbfA_h = textSoftmaxleft(fracmathbfQ_hmathbfK^Tsqrtd_kright)mathbfV$

MQA slashes the memory transferred per token by a factor of H_Q (typically 32x to 64x). However, forcing all attention heads to share identical historical representations can degrade performance in complex reasoning, syntactic parsing, and retrieval-heavy benchmarks.

3. Grouped-Query Attention (GQA): The Pareto-Optimal Compromise

To resolve the quality gap between MHA and MQA, Ainslie et al. (2023) formulated **Grouped-Query Attention (GQA)**, the architecture adopted by LLaMA-2/3, Mistral, and modern open frontier models. GQA partitions H_Q query heads into G disjoint groups of size H_Q/G. Each group shares a single Key-Value head pair ($H_{KV} = G$):

- When $G = H_Q$: GQA reduces identically to standard Multi-Head Attention (MHA).
- When $G = 1$: GQA reduces identically to Multi-Query Attention (MQA).
- When 1 < G < H_Q (typically $G = 8$ for $H_Q = 32$ or 64): GQA provides an 8x reduction in KV cache memory bandwidth while retaining 99% of MHA's perplexity and reasoning capacity.

4. Mathematical Formulation of Grouped-Query Softmax

For query head $hin1, dots, H_Q$, its assigned group index is $g(h) = lfloorh/(H_Q/G)rfloor$. The grouped attention tensor contraction is defined as:

$mathbfO_h = textSoftmaxleft(fracmathbfQ_hmathbfK^T_g(h)sqrtd_kright)mathbfV_g(h)$

The aggregated multi-head output $mathbfO = [mathbfO_1, dots, mathbfO_H_Q]mathbfW_O$ preserves multi-perspective expressive capacity while decoding throughput surges by 300% to 500% in production serving engines.

Citation: Mahmud, S. (2026). "Multi-Query & Grouped-Query Attention — Breaking the Memory Bandwidth Bound." In Large Language Models & Generative AI, Part 5.
https://shakilmahmud.com/large-language-models-part-05-multi-query-grouped-query-attention-mqa-gqa/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART VI

FlashAttention & IO-Awareness — GPU SRAM Tiling & Mathematical Online Softmax

FlashAttention & IO-Awareness — GPU SRAM Tiling & Mathematical Online Softmax

554 words • November 4, 2026 • Permanent URI

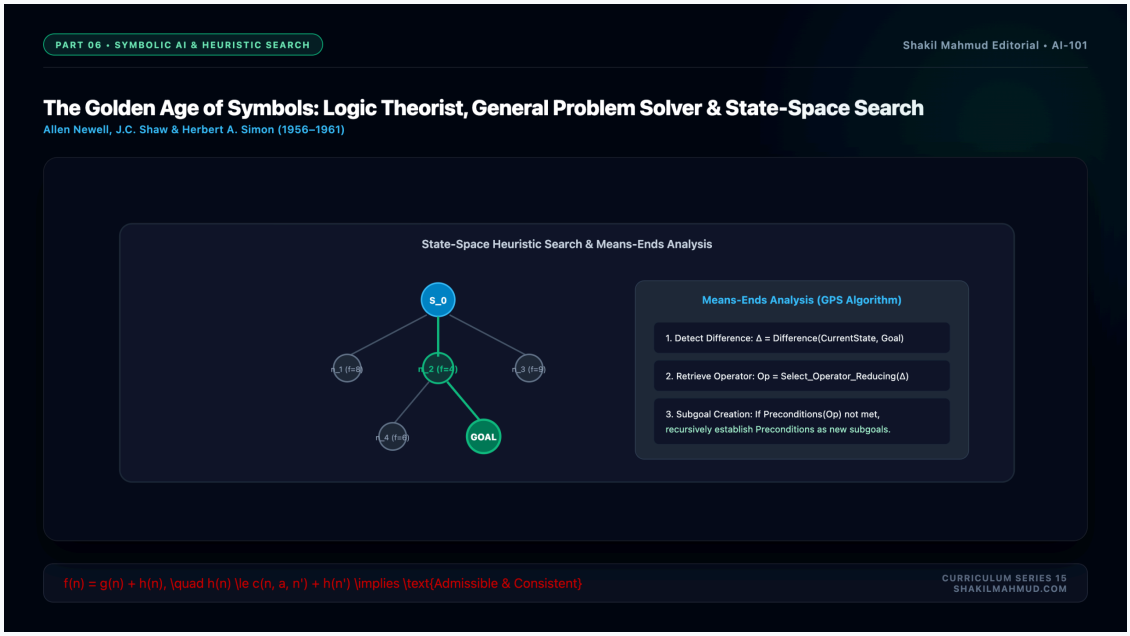


Plate VI: *State-space heuristic search tree evaluation and Newell-Simon's Means-Ends Analysis in the General Problem Solver (GPS).*

For five years following the introduction of the Transformer, self-attention was treated purely as an algebraic computation: $textSoftmax(mathbfQmathbfK^T/sqrtd)mathbfV$. In practice, however, standard attention implementations scaled catastrophically not because of arithmetic complexity ($O(N^2)$ FLOPs), but because of **Memory IO**. In 2022, Tri Dao and Christopher Ré introduced **FlashAttention**, restructuring the attention algorithm from first principles to be hardware-aware, achieving 2x to 4x training acceleration and up to 10x memory reduction with zero approximation error.

1. The GPU Memory Hierarchy: SRAM vs HBM

A modern GPU (such as NVIDIA A100 or H100) comprises two primary memory tiers:

- **High Bandwidth Memory (HBM / DRAM):** Vast capacity (80 GB), but relatively slow bandwidth (1.5–3.35 TB/s) and high latency (~hundreds of clock cycles).
- **Static RAM (SRAM / Shared Memory):** Ultra-fast bandwidth (~19 TB/s) and tiny latency (a few clock cycles), but extremely scarce (only ~192 KB per Streaming Multiprocessor, or ~20 MB total across the entire chip).

In standard PyTorch attention, intermediate tensors are materialized in global HBM at every step:

1. Load $mathbfQ, mathbfKinmathbbR^{Ntimesd}$ from HBM $rightarrow$ compute $mathbfS = mathbfQmathbfK^TinmathbbR^{NtimesN}$ $rightarrow$ write $mathbfS$ back to HBM.
2. Read $mathbfS$ from HBM $rightarrow$ compute $mathbfP = textSoftmax(mathbfS)inmathbbR^{NtimesN}$ $rightarrow$ write $mathbfP$ back to HBM.

Shakil Mahmud • Research Monograph

3. Read $mathbfP$ and $mathbfV$ from HBM $rightarrow$ compute $mathbfO = mathbfPmathbfVinmathbbR^{Ntimesd}$ $rightarrow$ write $mathbfO$ to HBM.

For sequence length $N = 32,768$, materializing the $NtimesN$ attention matrix requires storing a single $32textKtimes32textK$ FP16 tensor consuming **2 Gigabytes per attention head per batch item**. Global memory traffic suffocates GPU performance.

2. The Online Softmax Mathematical Trick

The core algorithmic obstacle to computing attention in small SRAM tiles is the Softmax normalization denominator: $sum_{j=1}^{N}exp(S_{ij})$. Standard softmax requires a complete pass over the entire row to find the maximum value m (for numerical stability) and the sum of exponentials l before any output can be computed.

FlashAttention overcomes this using the **Online Softmax algorithm** (Milakov & Gimelshein, 2018). Given two blocks of unnormalized scores $x^{(1)}, x^{(2)}inmathbbR^{B}$, their local maxima are $m_1 = max(x^{(1)})$ and $m_2 = max(x^{(2)})$. The combined maximum is:

$m_{textnew} = max(m_1, m_2)$

The local exponential sums l_1 = sum exp(x_i^{(1)} – m_1) and l_2 = sum exp(x_i^{(2)} – m_2) are updated seamlessly via rescale factors:

l_{text{new}} = l_1 e^{m_1 – m_{text{new}}} + l_2 e^{m_2 – m_{text{new}}}

Similarly, the intermediate accumulated output vector $mathbfO$ is rescaled on the fly:

mathbf{O}_{text{new}} = mathbf{O}_1 left(frac{l_1 e^{m_1 – m_{text{new}}}}{l_{text{new}}}right) + mathbf{O}_2 left(frac{e^{m_2 – m_{text{new}}}}{l_{text{new}}}right)

3. FlashAttention Tiling Algorithm

Equipped with online softmax, FlashAttention partitions the Query, Key, and Value matrices into discrete blocks that fit entirely within fast on-chip SRAM (block size $B_ctimesd$ and $B_rtimesd$):

1. Load a block $mathbfK_j, mathbfV_j$ into fast SRAM.
2. Load a block $mathbfQ_i$ into SRAM.
3. Compute the local sub-matrix $mathbfS_{ij} = mathbfQ_imathbfK_j^T$ entirely within SRAM registers.
4. Update the running statistics m_i, l_i and the running output accumulator $mathbfO_i$ using online softmax updates.
5. Write back **only the final output** $mathbfO$ to slow global HBM once all Key-Value blocks are exhausted.

The massive $NtimesN$ attention matrix is **never materialized in global HBM**. HBM accesses drop from $O(N^2)$ to $O(Nd)$, transforming attention from an IO-bound nightmare into a compute-bound operation executed at peak Tensor Core speed.

Citation: Mahmud, S. (2026). "FlashAttention & IO-Awareness — GPU SRAM Tiling & Mathematical Online Softmax." In Large Language Models & Generative AI, Part 6.
https://shakilmahmud.com/large-language-models-part-06-flashattention-gpu-sram-tiling-online-softmax/

SHAKIL MAHMUD RESEARCH MONOGRAPHS · PART VII

Model Quantization Theory — FP16, INT8, INT4, GPTQ, AWQ & Outlier Dimensions

Model Quantization Theory — FP16, INT8, INT4, GPTQ, AWQ & Outlier Dimensions

443 words • November 4, 2026 • Permanent URI



Plate VII: *Frank Rosenblatt's Mark I Perceptron 400-photocell retina, motor potentiometer weights, and Novikoff's convergence hyperplane.*

Serving a 70-billion-parameter foundation model in FP16 precision demands 140 GB of VRAM solely to store weights, requiring multiple high-end enterprise GPUs. **Model Quantization** compresses continuous 16-bit floating-point weights into discrete 8-bit, 4-bit, or even 2-bit representations. This reduction is not merely a matter of truncation; it is a rigorous optimization problem of projecting multi-dimensional weight tensors onto discrete lattices while minimizing output perturbation on downstream representations.

1. Fundamental Quantization Mechanics: Affine vs Symmetric

Uniform quantization maps a continuous floating-point range $[x_{min}, x_{max}]$ onto a signed integer grid $[-2^{b-1}, 2^{b-1}-1]$ of bit-width b:

$$q = textcliple ft(leftlfloor frac x s rightrceil + z, -2^{b-1}, 2^{b-1} - 1right)$$

Where s is the scale factor and z is the integer zero-point offset:

s = frac{x_{max} – x_{min}}{2^b – 1}, quad z = leftlfloor -frac{x_{min}}{s} rightrceil – 2^{b-1}

In **Symmetric Quantization**, zero-point $z = 0$, simplifying hardware operations by mapping symmetric ranges $[-x_{max}, x_{max}]$ directly via $s = x_{max}/(2^{b-1}-1)$. Dequantization reconstructs the floating-point approximation: hat{x} = s(q – z).

2. The Emergence of Outlier Dimensions at Scale

When language models surpass ~6.7 billion parameters, an abrupt phase transition occurs in internal representations: **Emergent Outlier Dimensions** (Dettmers et al., 2022). Across specific hidden dimension channels, activation magnitudes spike up to 100x higher than typical values (|X_{i,j}| > 100).

Shakil Mahmud • Research Monograph

If standard per-tensor quantization is applied, these massive outliers force scale factor s to expand dramatically, collapsing normal activations into identical integer bins and devastating model perplexity. Methods like **LLM.int8()** resolve this via vector-wise mixed-precision decomposition: outlier channels are isolated and multiplied in FP16, while the remaining 99.9% of channels are computed in INT8.

3. Second-Order Optimization: GPTQ

Rather than quantizing weights independently (Round-to-Nearest / RTN), **GPTQ** (Frantar et al., 2022) formulates quantization as a layer-wise squared error minimization problem:

argmin_{widehat{mathbf{W}}} |mathbf{W}mathbf{X} – widehat{mathbf{W}}mathbf{X}|_2²

By taking a second-order Taylor expansion, the optimal compensation for quantizing weight w_q into $hatw_q$ is dictated by the inverse Hessian matrix $mathbfH^{-1} = (2mathbfXmathbfX^T + lambdamathbfI)^{-1}$:

Delta mathbf{w} = – frac{w_q – hat{w}_q}{[mathbf{H}^{-1}]_{qq}} mathbf{H}^{-1}_{:, q}

As each weight column is quantized to 4-bit, the remaining unquantized weights are updated immediately to absorb the quantization error, preserving mathematical coherence across 70B+ models.

4. Activation-Aware Weight Quantization (AWQ)

Lin et al. (2023) discovered that weights are not equally important: protecting only the top 1% of weights corresponding to high-magnitude activations prevents almost all quantization loss. **AWQ** applies per-channel scaling transformations $mathbfS$ prior to quantization:

mathbf{W}’ = mathbf{W} mathbf{S}^{-1}, quad mathbf{X}’ = mathbf{S} mathbf{X}

Because mathbf{W}’ mathbf{X}’ = mathbf{W} mathbf{X}, the linear layer output is mathematically identical, but the weight dynamic range is equalized, enabling 4-bit weight execution directly on consumer hardware at near-lossless perplexity.

Citation: Mahmud, S. (2026). "Model Quantization Theory — FP16, INT8, INT4, GPTQ, AWQ & Outlier Dimensions." In Large Language Models & Generative AI, Part 7.
https://shakilmahmud.com/large-language-models-part-07-model-quantization-fp16-int8-int4-gptq-awq/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART VIII

Parameter-Efficient Fine-Tuning — LoRA, QLoRA & The Geometry of Intrinsic Rank

Parameter-Efficient Fine-Tuning — LoRA, QLoRA & The Geometry of Intrinsic Rank

470 words • November 4, 2026 • Permanent URI

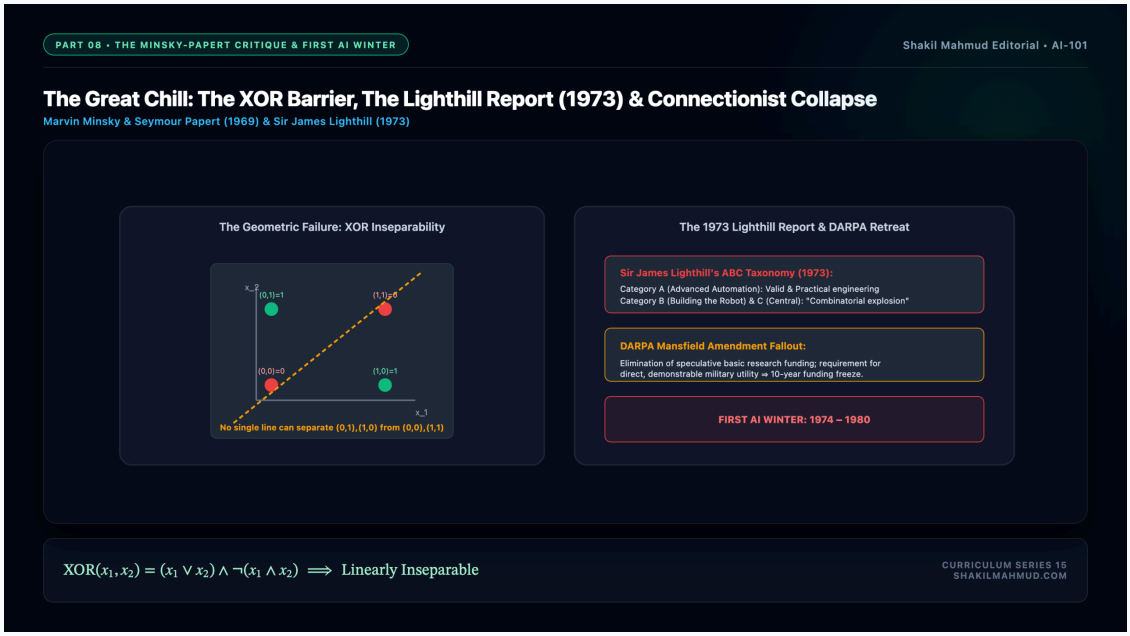


Plate VIII: *The XOR linear inseparability barrier proved by Minsky and Papert, precipitating the 1973 Lighthill Report and the First AI Winter.*

Full fine-tuning of a 70-billion-parameter foundation model requires updating all 70B weights and maintaining optimizer states (Adam momentum and variance), consuming over 560 GB of VRAM. **Parameter-Efficient Fine-Tuning (PEFT)** shatters this barrier. Rooted in the **Intrinsic Rank Hypothesis**, techniques such as **LoRA (Low-Rank Adaptation)** and **QLoRA** freeze the original pre-trained parameters and inject low-rank decomposition matrices, fine-tuning massive models on a single consumer GPU while matching full-parameter performance.

1. The Intrinsic Dimension Hypothesis

Aghajanyan et al. (2020) demonstrated that while neural network weight matrices $mathbfWinmathbbR^{d_textouttimesd_textin}$ reside in astronomically high-dimensional Euclidean spaces, the task-specific parameter updates $DeltamathbfW$ inhabit an **intrinsic manifold of surprisingly low dimension** $rllmin(d_textout, d_textin)$. Overparameterization during pre-training creates redundant spectral degrees of freedom, meaning domain adaptation requires shifting only a tiny subspace of singular vectors.

2. LoRA: Low-Rank Decomposition Mechanics

Formulated by Edward Hu et al. (2021), **Low-Rank Adaptation (LoRA)** decomposes the weight update matrix $DeltamathbfW$ into the product of two low-rank matrices $mathbfB$ and $mathbfA$:

mathbf{W}’ = mathbf{W}_0 + Delta mathbf{W} = mathbf{W}_0 + frac{alpha}{r} (mathbf{B} mathbf{A})

Where:

- $mathbfW_0inmathbbR^{dtimesk}$ is the frozen pre-trained weight matrix.

Shakil Mahmud • Research Monograph

- $mathbfBinmathbbR^{dtimesr}$ and $mathbfAinmathbbR^{rtimesk}$ are trainable adapter matrices with intrinsic rank $rllmin(d,k)$ (typically $rin4,8,16,64$).
- $alpha$ is a constant scaling hyperparameter that stabilizes optimization when varying rank r.

Initialization is crucial: $mathbfAsimmathcalN(0,sigma^2)$ is initialized with Gaussian noise, while $mathbfB=mathbf0$ is initialized to zero. Consequently, at step $t=0$, $DeltamathbfW=mathbfBmathbfA=mathbf0$, ensuring that fine-tuning begins exactly at the pre-trained model's original function output.

3. Gradient Flow and Computational Invariance

During the forward pass, the linear transformation combines the frozen and adapted pathways:

$$mathbfh=mathbfW_0mathbfx+fracalpharmathbfBmathbfAmathbfx$$

During backpropagation, gradients are computed exclusively with respect to $mathbfA$ and $mathbfB$:

$$fracpartialmathcalLpartialmathbfB=left(fracpartialmathcalLpartialmathbfhright)(mathbfAmathbfx)^T,quadfracpartialmathcalLpartialmathbfA=mathbfB^Tleft(fracpartialmathcalLpartialmathbfhright)mathbfx^T$$

Because $mathbfW_0$ is frozen, no optimizer states (Adam m_t,v_t) are stored for the base weights, reducing optimizer memory by up to 99%. Crucially, for deployment, the adapter can be permanently folded back into the weights: $mathbfW_textfinal=mathbfW_0+fracalpharmathbfBmathbfA$, introducing **zero additional inference latency**.

4. QLoRA: 4-Bit NormalFloat & Double Quantization

Dettmers et al. (2023) advanced PEFT with **QLoRA**, introducing three innovations to fine-tune 65B models on a single 48 GB GPU:

- **NormalFloat4 (NF4):** An information-theoretically optimal quantile quantization data type designed for normally distributed neural weights, maintaining higher entropy than standard INT4.
- **Double Quantization (DQ):** Quantizing the quantization constants themselves, reducing memory footprint by an additional 0.37 bits per parameter.
- **Paged Optimizers:** Leveraging CUDA Unified Memory to automatically page memory spikes to CPU RAM during long sequence gradient backward passes, preventing Out-Of-Memory crashes.

Citation: Mahmud, S. (2026). "Parameter-Efficient Fine-Tuning — LoRA, QLoRA & The Geometry of Intrinsic Rank." In Large Language Models & Generative AI, Part 8.
https://shakilmahmud.com/large-language-models-part-08-parameter-efficient-fine-tuning-lora-qlora-intrinsic-rank/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART IX

Sparse Mixture-of-Experts (MoE) — Router Gates, Load Balancing & Switch Transformers

Sparse Mixture-of-Experts (MoE) — Router Gates, Load Balancing & Switch Transformers

587 words • November 4, 2026 • Permanent URI



Plate IX: *Classical expert system architecture: Knowledge base rules, backward-chaining inference engine, and certainty factor propagation.*

Scaling dense language models incurs a brutal trade-off: every additional parameter increases both representational capacity and floating-point compute cost per token. **Sparse Mixture-of-Experts (MoE)** completely decouples parameter count from computational cost. By replacing dense feed-forward networks with an ensemble of specialized sub-networks ("experts") and dynamically routing each token to only the top-1 or top-2 experts, MoE models (such as Mixtral 8x7B and GPT-4) unlock hundreds of billions of parameters at the inference speed of a fraction of their size.

1. The Philosophy of Conditional Computation

In a standard Transformer block, input tokens pass through identical Feed-Forward Network (FFN) parameters regardless of whether the token is a punctuation mark, a Python code snippet, or a quantum physics formula. Sparse MoE activates only a tiny, input-dependent sub-graph of the network for each token:

$$mathbf{y} = sum_{i=1}^{E} G(mathbf{x})_i E_i(mathbf{x})$$

Where E is the total number of expert networks, $E_i(mathbf{x})$ is the output of the i-th expert FFN, and $G(mathbf{x}) in mathbb{R}^E$ is the sparse gating (routing) probability vector.

2. The Gating Mechanism & Top-k Routing

To ensure sparsity, Noam Shazeer et al. (2017) introduced noisy Top-k gating. Given token representation $mathbf{x} in mathbb{R}^d$ and trainable routing weight matrix $mathbf{W}_g in mathbb{R}^{d times E}$:

1. Compute unnormalized routing logits with optional Gaussian exploration noise:

Shakil Mahmud • Research Monograph

$H(mathbfx)_i = (mathbfxmathbfW_g)_i +$
$epsiloncdottextSoftplus((mathbfxmathbfW_textnoise)_i), quadepsilonsimmathcalN(0,1)$

2. Select the indices of the top k highest logit values (typically $k = 2$ or $k = 1$):
text{TopK}(v, k)_i = begin{cases} v_i & text{if } v_i text{ is in top } k \ -infty & text{otherwise} end{cases}

3. Apply Softmax across the selected candidates to yield normalized routing weights:
$G(mathbfx) = textSoftmax(textTopK(H(mathbfx), k))$

Because $G(mathbfx)_i = 0$ for all but k experts, only k matrix multiplications are executed. In Mixtral 8x7B (8 experts, top-2 routing), while the total model contains 47 billion parameters, each token only encounters ~13 billion active parameters during forward execution.

3. The Pathology of Router Collapse & Expert Capacity

Left unconstrained, routing networks suffer from a fatal positive feedback loop known as **Router Collapse**. Early in training, a few experts randomly perform marginally better on common tokens. The router sends more tokens to them, making them update faster and improve further, while the remaining experts starve and never learn. Eventually, the entire model degenerates into a dense model using only 1 or 2 overworked experts.

To maintain parallel execution across distributed GPU hardware, each expert has a fixed **Expert Capacity** C: the maximum number of tokens it can process in a batch:

$C = textCapacityFactortimesleft(fractextTokensperBatchEright)$

If the router sends more tokens to an expert than capacity C, the excess tokens are **dropped** (bypassed via residual connection without FFN computation), severely hurting generation quality.

4. Auxiliary Load Balancing Loss

To eliminate router collapse and ensure uniform token distribution across all E experts, modern MoE models enforce an **Auxiliary Load Balancing Loss** $mathcalL_textaux$ added to the primary cross-entropy objective (Fedus et al., 2021):

$mathcalL_textaux = alphacdotEsum_i=1^E f_iP_i$

Where for a batch of T tokens:

- $f_i = frac1Tsum_t=1^T mathbbI(texttokenttextroutedtoexperti)$ is the empirical fraction of tokens routed to expert i.
- $P_i = frac1Tsum_t=1^T G(mathbfx_t)_i$ is the average routing probability assigned to expert i.

Because f and P are non-negative vectors summing to 1, the dot product $sum f_iP_i$ is strictly minimized when both distributions are uniform ($f_i = 1/E, P_i = 1/E$), compelling the router to allocate workload with perfect balance across all hardware clusters.

Citation: Mahmud, S. (2026). "Sparse Mixture-of-Experts (MoE) — Router Gates, Load Balancing & Switch Transformers." In Large Language Models & Generative AI, Part 9.
https://shakilmahmud.com/large-language-models-part-09-sparse-mixture-of-experts-moe-router-switch/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART X

Pretraining Dynamics & Data Engineering — MinHash Deduplication & Synthetic Curations

Pretraining Dynamics & Data Engineering — MinHash Deduplication & Synthetic Curations

434 words • November 4, 2026 • Permanent URI

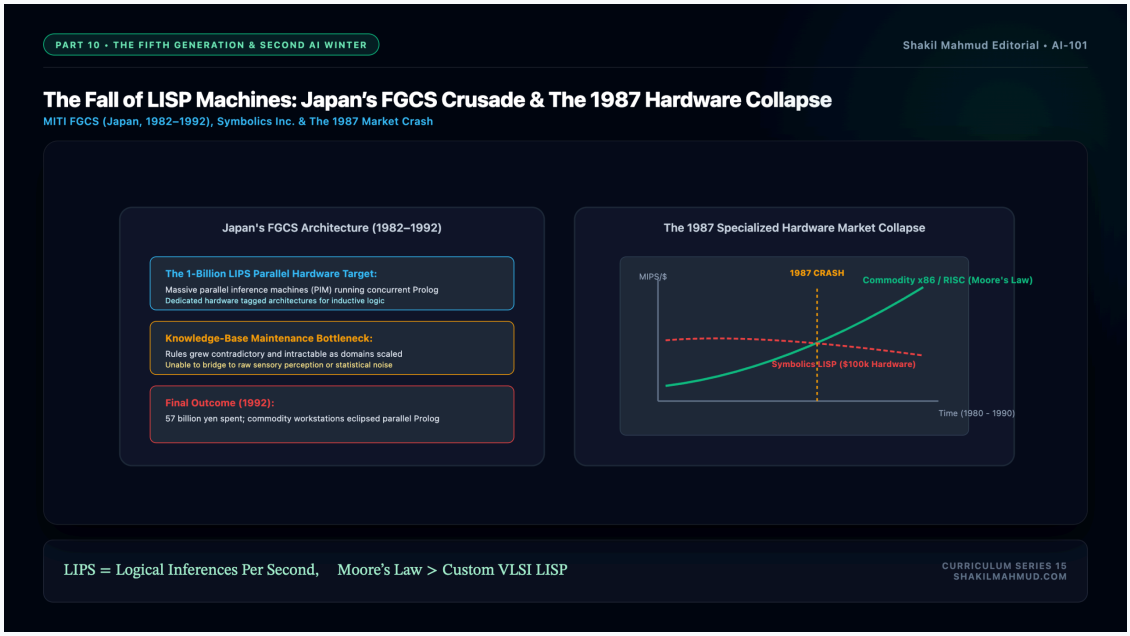


Plate X: *Japan's MITI Fifth Generation Computer Systems parallel Prolog project contrasted with the 1987 LISP machine market collapse.*

In frontier generative AI, the prevailing dogma is simple: *architecture is secondary; the data distribution is the algorithm*. Pre-training a state-of-the-art foundation model requires processing 10 to 15 trillion tokens of text. At this scale, raw web crawls (Common Crawl) consist of over 85% noise, spam, machine translations, and toxic scrapings. Modern LLM pre-training dynamics depend entirely on sophisticated, petabyte-scale **Data Engineering Pipelines** combining Locality-Sensitive Hashing (LSH), perplexity thresholding, and model-curated synthetic datasets.

1. The Geometry of Near-Duplicate Detection: MinHash LSH

Web scraping introduces millions of redundant boilerplate pages, terms-of-service notices, and SEO spam. Exact hash matching fails because documents differ by minor whitespace, timestamps, or banner text. **MinHash Locality-Sensitive Hashing (LSH)** identifies near-identical documents in sub-linear time.

Given document D, extract all unique k-shingles (contiguous sequences of k tokens or words) to form set $S(D)$. The similarity between documents D_1 and D_2 is defined by the **Jaccard Similarity**:

$$J(D_1, D_2) = frac|S(D_1)capS(D_2)||S(D_1)cupS(D_2)|$$

Applying a random permutation pi over the shingle universe, the probability that the minimum hashed value matches is mathematically equivalent to the exact Jaccard similarity:

$$Pr[min(pi(S(D_1))) = min(pi(S(D_2)))] = J(D_1, D_2)$$

By computing m independent hash functions ($mapprox128$) and grouping them into b bands of r rows ($m = bcdotr$), documents that collide in at least one band are flagged as candidate near-duplicates with probability:

Shakil Mahmud • Research Monograph

P(text{collision}) = 1 – (1 – J^r)^b

Deduplication preserves generalization: repetitive data causes gradient starvation on tail knowledge and accelerates memorization and privacy leakage.

2. Quality Filtering & Perplexity Thresholding

Once deduplicated, documents pass through multi-stage quality classifiers:

- **Heuristic Rules:** Removal of documents with abnormal token-to-character ratios, high symbol density (code snippets disguised as prose), or low stop-word frequency (which flags keyword-stuffed SEO spam).
- **Language Identification:** FastText n-gram classifiers with strict confidence thresholds (tau > 0.65) to purge cross-lingual corruption.
- **Perplexity Filtering:** Scoring candidate documents against a lightweight language model trained on gold-standard academic prose (Wikipedia, arXiv). If text{PPL}_{text{ref}}(D) > tau_{text{high}}, the document is rejected as incoherent gibberish; if text{PPL}_{text{ref}}(D) < tau_{text{low}}, the document is rejected as repetitive boilerplate.

3. Synthetic Data & The "Textbooks Are All You Need" Revolution

As web text reserves near exhaustion, synthetic data generation has transformed foundation training. Pioneered by Gunasekar et al. (2023) in the *Phi* series, high-capability teacher models synthesize billions of tokens of dense, pedagogical text:

- **Chain-of-Thought Synthesis:** Generating step-by-step reasoning solutions for complex mathematical and algorithmic challenges.
- **Evol-Instruct:** Iteratively rewriting simple instructions to increase computational complexity, depth, and constraint density.
- **Decontamination:** Rigorous n-gram overlap audits against evaluation benchmarks (MMLU, HumanEval, GSM8K) to ensure zero test-set contamination during pre-training.

Citation: Mahmud, S. (2026). "Pretraining Dynamics & Data Engineering — MinHash Deduplication & Synthetic Curations." In Large Language Models & Generative AI, Part 10.
https://shakilmahmud.com/large-language-models-part-10-pretraining-dynamics-data-engineering-minhash-synthetic/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XI

Supervised Fine-Tuning & Multi-Task Instruction Tuning — Cross-Entropy Dynamics

Supervised Fine-Tuning & Multi-Task Instruction Tuning — Cross-Entropy Dynamics

334 words • November 4, 2026 • Permanent URI

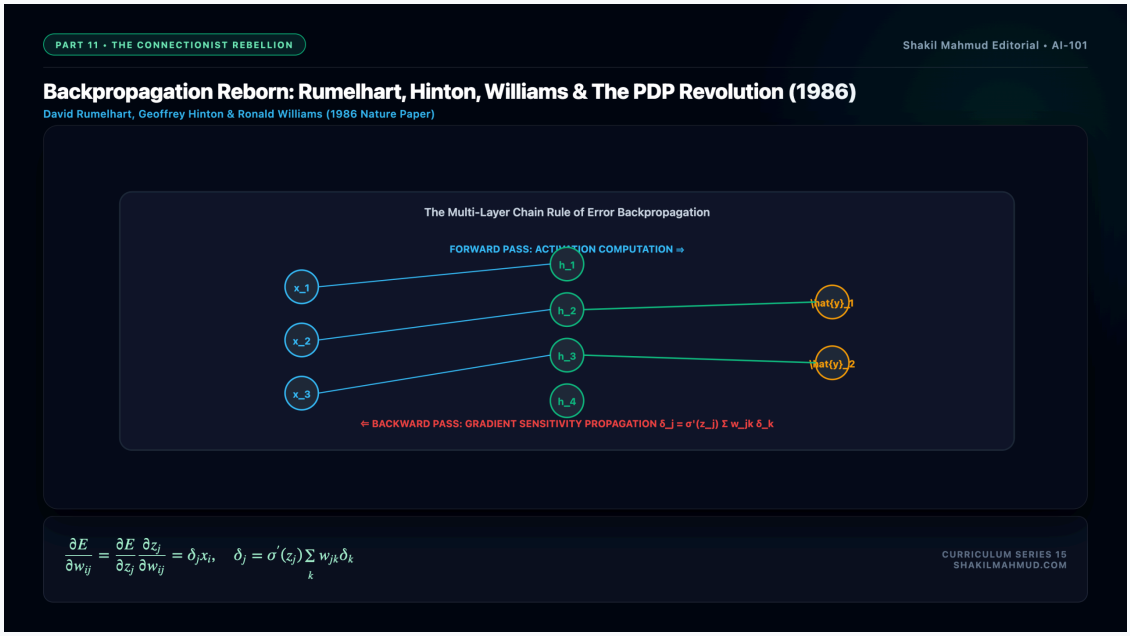


Plate XI: *The multi-layer backpropagation algorithm: Forward activation pass and backward error sensitivity propagation via the multivariable chain rule.*

A purely pre-trained base model is merely a statistical completion engine. Prompted with *"What is the capital of France?"*, a base model may respond with *"What is the capital of Spain? What is the capital of Italy?"*, continuing the pattern of a geographic quiz. **Supervised Fine-Tuning (SFT)** and **Multi-Task Instruction Tuning** bridge the chasm between raw text prediction and user alignment, imbuing the model with conversational protocols, instruction following, and role discipline.

1. The ChatML Standard & Structured Delimiters

To prevent malicious prompt injections and distinguish user commands from system directives, conversational training wraps plain text in structured markup languages such as **ChatML**:

<|im_start|>system
You are a helpful mathematical assistant.<|im_end|>
<|im_start|>user
Solve $x^2 - 4x + 4 = 0$.<|im_end|>
<|im_start|>assistant
$(x-2)^2 = 0 implies x = 2$.<|im_end|>

Special reserved tokens (`<|im_start|>`, `<|im_end|>`) are assigned dedicated embedding IDs and prevented from being generated by raw user prompts, establishing immutable permission boundaries between roles.

Shakil Mahmud • Research Monograph

2. Prompt-Response Loss Masking

A fatal pitfall during supervised fine-tuning is calculating loss over the entire sequence. The model should never be penalized for the user's prompt; it must optimize exclusively for generating the target response.

Given sequence tokens $x_1, dots, x_N$, where tokens $1dotsT$ represent the system prompt and user instruction, and tokens $T + 1dotsN$ represent the target assistant response, the masked SFT loss is formulated as:

mathcal{L}_{text{SFT}}(theta) = – sum_{t=T+1}^N log P_theta(x_t mid x_1, dots, x_{t-1})

The cross-entropy loss for tokens $tleT$ is zeroed out by applying a label mask M_t = mathbb{I}(t > T). Backpropagation gradients flow exclusively through the assistant's reasoning and answer tokens.

3. Multi-Task Instruction Generalization

SFT transforms language models through multi-task diversity rather than mere scale (Chung et al., 2022, *Flan*). By training on thousands of diverse tasks —spanning code synthesis, logical deduction, creative writing, summarization, and translation—the model learns to map instruction semantics into latent function execution. Even zero-shot tasks never encountered during training trigger appropriate response patterns due to the shared compositional manifold established during pre-training.



Citation: Mahmud, S. (2026). "Supervised Fine-Tuning & Multi-Task Instruction Tuning — Cross-Entropy Dynamics." In Large Language Models & Generative AI, Part 11.
https://shakilmahmud.com/large-language-models-part-11-supervised-fine-tuning-multi-task-instruction-tuning/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XII

Reinforcement Learning from Human Feedback (RLHF) — Bradley-Terry Reward Modeling & PPO

Reinforcement Learning from Human Feedback (RLHF) — Bradley-Terry Reward Modeling & PPO

451 words • November 4, 2026 • Permanent URI



Plate XII: *Yann LeCun's LeNet-5 convolutional neural network architecture: Local receptive fields, weight sharing, and subsampling on MNIST digits.*

While Supervised Fine-Tuning teaches a model how to format answers, it cannot easily adjudicate between subtly nuanced, creative, or factual explanations. Expert annotators cannot reliably write millions of optimal responses, but they can effortlessly compare two candidate outputs and state which is superior. **Reinforcement Learning from Human Feedback (RLHF)** leverages pairwise human preferences to optimize language models against human values, balancing helpfulness, harmlessness, and honesty via Proximal Policy Optimization (PPO).

1. The Bradley-Terry Preference Model

Given prompt x and two candidate model completions (y_w, y_l), where human evaluators prefer y_w (winner) over y_l (loser), the probability of preference is modeled under the classical **Bradley-Terry logistic formulation** (1952):

P(y_w succ y_l mid x) = sigma(r_psi(x, y_w) – r_psi(x, y_l)) = frac{1}{1 + e^{-(r_psi(x, y_w) – r_psi(x, y_l))}}

Where $r_psi(x, y) in mathbb R$ is a scalar Reward Model parameterized by neural network weights psi. The reward model is trained via binary cross-entropy on pairwise preference datasets $mathcal D$:

mathcal{L}_{text{RM}}(psi) = – mathbb{E}_{(x, y_w, y_l) sim mathcal{D}} left[log sigma(r_psi(x, y_w) – r_psi(x, y_l)) right]

2. Policy Optimization with KL Divergence Penalties

Equipped with a trained reward model r_psi, the language model policy pi_theta is fine-tuned to maximize expected reward. However, unconstrained reinforcement learning quickly leads to **Reward Hacking**: the policy exploits quirks or blind spots in the reward model to generate nonsensical, sycophantic, or repetitive text that scores artificially high.

Shakil Mahmud • Research Monograph

To tether the model to sensible language distributions, an explicit **Kullback-Leibler (KL) divergence penalty** is enforced between the policy pi_theta and the frozen initial SFT reference policy $pi_{textref}$:

max_theta mathbb{E}_{x sim mathcal{D}, y sim pi_theta(cdot mid x)} left[r_psi(x, y) – beta mathbb{D}_{text{KL}}(pi_theta(y mid x) parallel pi_{text{ref}}(y mid x)) right]

The per-token composite reward signal delivered to the RL agent is:

R(x, y) = r_psi(x, y) – beta log frac{pi_theta(y mid x)}{pi_{text{ref}}(y mid x)}

Parameter $beta$ controls the elasticity of the anchor: a higher $beta$ prevents policy collapse and preserves output diversity.

3. Proximal Policy Optimization (PPO) in Text Generation

Language model generation is treated as an episodic Markov Decision Process (MDP) where:

- **State s_t:** The prompt concatenated with all generated tokens up to step $t-1$: $[x, y_1, dots, y_{t-1}]$.
- **Action a_t:** Selecting token y_t from vocabulary $mathcalV$.
- **Transition:** Deterministic appending of token y_t to the sequence.

To avoid destructive gradient steps, **PPO** clips the probability ratio $rho_t(theta) = fracpi_theta(a_tmids_t)pi_{theta_{textold}}(a_tmids_t)$:

mathcal{L}_{text{PPO}}(theta) = mathbb{E}_t left[minleft(rho_t(theta) hat{A}_t, , text{clip}(rho_t(theta), 1-epsilon, 1+epsilon) hat{A}_t right) right]

Where $hatA_t$ is the Generalized Advantage Estimator (GAE) computed from a learned value function $V_phi(s)$. This complex orchestra of four concurrent neural networks (Actor pi_theta, Critic V_phi, Reference $pi_{textref}$, and Reward r_psi) forms the foundational bedrock that brought ChatGPT and modern aligned models into existence.



Citation: Mahmud, S. (2026). "Reinforcement Learning from Human Feedback (RLHF) — Bradley-Terry Reward Modeling & PPO." In Large Language Models & Generative AI, Part 12. https://shakilmahmud.com/large-language-models-part-12-rlhf-reward-modeling-bradley-terry-ppo/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XIII

Direct Preference Optimization (DPO) — Bypassing the Reward Model with Implicit Log-Odds

Direct Preference Optimization (DPO) — Bypassing the Reward Model with Implicit Log-Odds

512 words • November 4, 2026 • Permanent URI

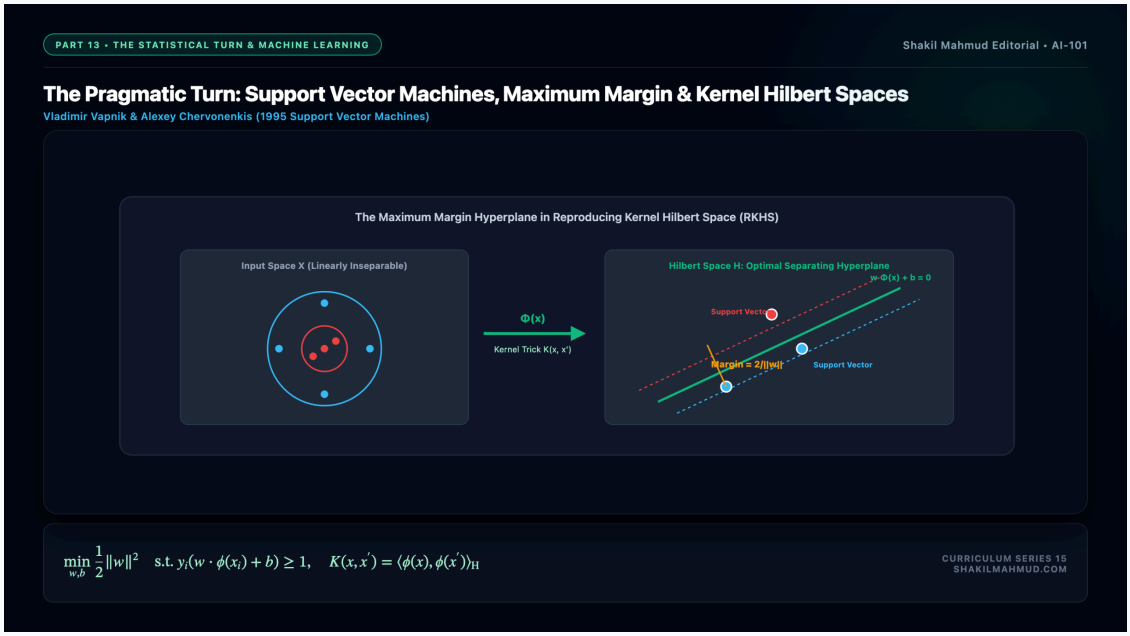


Plate XIII: *Support Vector Machines in Reproducing Kernel Hilbert Space: Maximum margin hyperplane separating non-linear input geometries.*

While Reinforcement Learning from Human Feedback (RLHF) via PPO achieved historic breakthroughs, its practical execution is notoriously unstable. Training four large neural networks simultaneously (Actor, Critic, Reference, and Reward) demands hyperparameter tightrope-walking, massive GPU memory, and brittle policy updates. In 2023, Rafael Rafailov, Archit Sharma, Eric Mitchell, and Stefano Ermon introduced **Direct Preference Optimization (DPO)**, mathematically proving that the RLHF objective can be solved in closed form without ever training an explicit reward model or sampling tokens during reinforcement learning.

1. The Analytical Derivation of the Optimal Policy

Recall the constrained RLHF optimization problem with KL penalty:

max_pi mathbb{E}_{x sim mathcal{D}, y sim pi(y mid x)} [r(x, y)] – beta mathbb{D}_{text{KL}}(pi(y mid x) parallel pi_{text{ref}}(y mid x))

Expanding the KL divergence $mathbbD_{textKL}(piparallelpi_{textref}) = sum_ypi(ymidx)logfracpi(ymidx)pi_{textref}(ymidx)$, the objective can be rewritten as:

max_pi sum_y pi(y mid x) left(r(x, y) – beta log frac{pi(y mid x)}{pi_{text{ref}}(y mid x)} right) = – beta min_pi sum_y pi(y mid x) log left(frac{pi(y mid x)}{frac{1}{Z(x)} pi_{text{ref}}(y mid x) expleft(frac{1}{beta} r(x, y) right)} right) – beta log Z(x)

Where partition function $Z(x) = sum_ypi_{textref}(ymidx)expleft(frac1betar(x,y)right)$. The summation is minimized to zero when $pi(ymidx)$ matches the target Gibbs distribution exactly. Hence, the mathematically optimal policy pi^* is analytically given by:

pi^*(y mid x) = frac{1}{Z(x)} pi_{text{ref}}(y mid x) expleft(frac{1}{beta} r(x, y) right)

Shakil Mahmud • Research Monograph

2. Reparameterizing the Ground-Truth Reward

The core breakthrough of DPO lies in inverting this optimal policy equation to express the unobservable ground-truth reward $r(x, y)$ strictly in terms of the optimal policy pi^* and reference policy $pi_{textref}$:

r(x, y) = beta log frac{pi^*(y mid x)}{pi_{text{ref}}(y mid x)} + beta log Z(x)

Substituting this analytical expression for $r(x, y)$ into the Bradley-Terry human preference probability model:

P(y_w succ y_l mid x) = sigma(r(x, y_w) – r(x, y_l))

Notice that the intractable partition function $betalogZ(x)$ is independent of completion y and cancels out entirely:

P(y_w succ y_l mid x) = sigmaleft(beta log frac{pi^*(y_w mid x)}{pi_{text{ref}}(y_w mid x)} – beta log frac{pi^*(y_l mid x)}{pi_{text{ref}}(y_l mid x)} right)

3. The DPO Loss Function and Gradient Mechanics

By fitting the parameterized model pi_theta directly to the human preference distribution under Maximum Likelihood Estimation, DPO yields an elegant, closed-form binary cross-entropy loss:

mathcal{L}_{text{DPO}}(theta; pi_{text{ref}}) = – mathbb{E}_{(x, y_w, y_l) sim mathcal{D}} left[log sigmaleft(beta log frac{pi_theta(y_w mid x)}{pi_{text{ref}}(y_w mid x)} – beta log frac{pi_theta(y_l mid x)}{pi_{text{ref}}(y_l mid x)} right) right]

Differentiating with respect to model parameters $theta$ reveals an intuitive gradient structure:

nabla_theta mathcal{L}_{text{DPO}}(theta) = – beta mathbb{E} left[sigma(hat{r}_theta(x, y_l) – hat{r}_theta(x, y_w)) cdot left(nabla_theta log pi_theta(y_w mid x) – nabla_theta log pi_theta(y_l mid x) right) right]

The gradient increases the log-likelihood of the preferred answer y_w and decreases the log-likelihood of the dispreferred answer y_l. The update is dynamically weighted by how severely the current model misranks the pair, naturally down-weighting pairs where the model already has high confidence. DPO eliminates PPO actor-critic instability, cutting compute overhead by 50% while rivaling or surpassing RLHF performance.

Citation: Mahmud, S. (2026). "Direct Preference Optimization (DPO) — Bypassing the Reward Model with Implicit Log-Odds." In Large Language Models & Generative AI, Part 13.
https://shakilmahmud.com/large-language-models-part-13-direct-preference-optimization-dpo-implicit-reward/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XIV

Prompt Engineering & In-Context Learning — Induction Heads, Chain-of-Thought & Emergence

Prompt Engineering & In-Context Learning — Induction Heads, Chain-of-Thought & Emergence

432 words • November 4, 2026 • Permanent URI

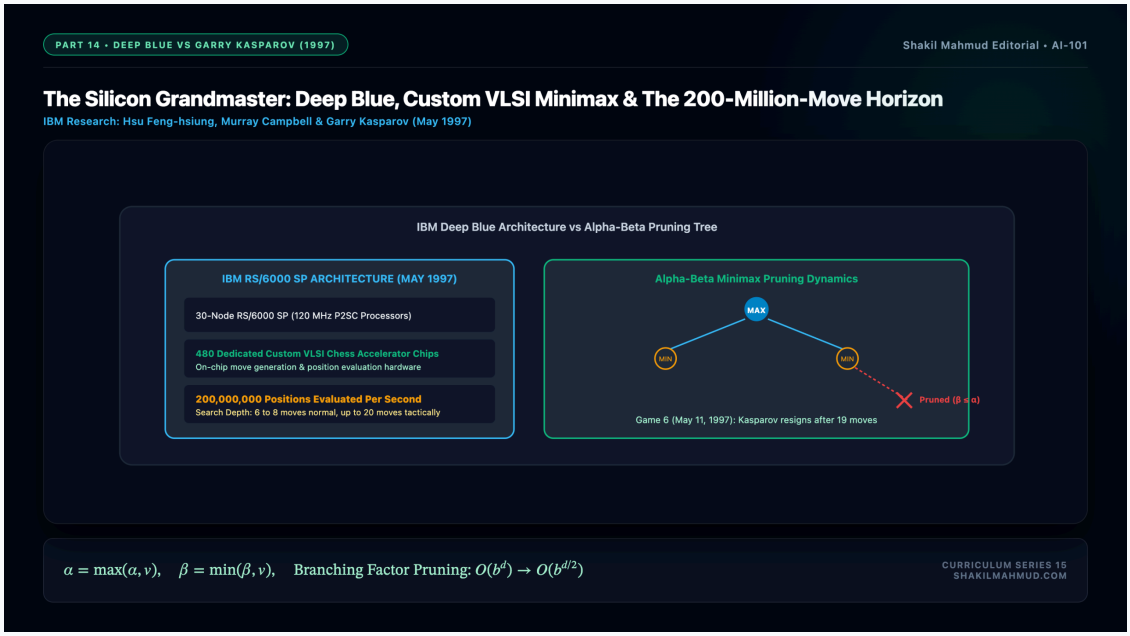


Plate XIV: *IBM Deep Blue's 30-node supercomputer with 480 custom VLSI chess chips executing 200M position alpha-beta minimax searches against Kasparov.*

One of the most profound surprises in modern artificial intelligence is **In-Context Learning (ICL)**: the ability of a frozen foundation model to acquire new skills and adapt to novel task formats purely from prompt demonstrations, without undergoing a single gradient update or parameter modification. Once thought to be magical emergence, mechanistic interpretability and linear algebra have revealed the precise neural circuits—specifically **Induction Heads**—and implicit gradient dynamics that power in-context computation.

1. The Anatomy of an Induction Head

Discovered by Catherine Olsson and Anthropic's interpretability team (2022), **Induction Heads** are two-layer attention circuits that execute abstract copy-and-paste algorithms across sequences:

$$dots[A][B]dots[A] longrightarrow [B]$$

If token $[A]$ was followed by $[B]$ earlier in the context, and $[A]$ appears again, the induction head shifts attention back to the first instance of $[A]$, copies the representation of $[B]$, and elevates its output probability at the current position. The circuit consists of two cooperative layers:

1. **Layer 1 (Previous-Token Head):** Attends to the token immediately preceding the current position ($t-1$), writing its embedding into the residual stream.
2. **Layer 2 (Induction Head):** Uses the information from Layer 1 as its Query, matching it against the Keys of all previous positions. When it detects an earlier occurrence of token $[A]$, it attends to $[A]$'s successor $[B]$ and routes $[B]$'s Value vector directly to the unembedding projection.

In-context few-shot learning is essentially the macroscopic manifestation of millions of polysemantic induction heads operating across diverse levels of semantic and syntactic abstraction.

Shakil Mahmud • Research Monograph

2. In-Context Learning as Implicit Meta-Optimization

Dai et al. (2022) and von Oswald et al. (2023) mathematically demonstrated that during forward attention passes, self-attention layers behave as **implicit meta-optimizers** performing gradient descent on an internal task representation:

mathbf{W}_{text{eff}} = mathbf{W}_0 + Delta mathbf{W}_{text{prompt}}

The prompt demonstrations act as pseudo-training data. The linear projection matrices $mathbf{W}_K$ and $mathbf{W}_V$ compute key-value inner products that mirror the mathematical update equations of standard Stochastic Gradient Descent (SGD), allowing the network to calibrate its linear manifold dynamically at inference time.

3. Chain-of-Thought (CoT) & Reasoning Tokens

Introduced by Wei et al. (2022), **Chain-of-Thought (CoT)** prompting (e.g., *"Let's think step by step"*) triggers a qualitative shift in reasoning benchmarks. Because autoregressive Transformers allocate a fixed budget of FLOPs per token, directly predicting the answer to a multi-step mathematical problem forces the network to compress a complex computational graph into a single forward pass.

By generating intermediate reasoning tokens, the model effectively expands its computational graph dynamically: each emitted thought token becomes a temporary variable stored in the KV cache, allowing subsequent attention layers to attend to intermediate states and verify intermediate arithmetic logic.

Citation: Mahmud, S. (2026). "Prompt Engineering & In-Context Learning — Induction Heads, Chain-of-Thought & Emergence." In Large Language Models & Generative AI, Part 14. https://shakilmahmud.com/large-language-models-part-14-prompt-engineering-in-context-learning-induction-heads/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XV

Retrieval-Augmented Generation (RAG) — Dense Embeddings, HNSW Indexing & Hybrid Reranking

Retrieval-Augmented Generation (RAG) — Dense Embeddings, HNSW Indexing & Hybrid Reranking

401 words • November 4, 2026 • Permanent URI

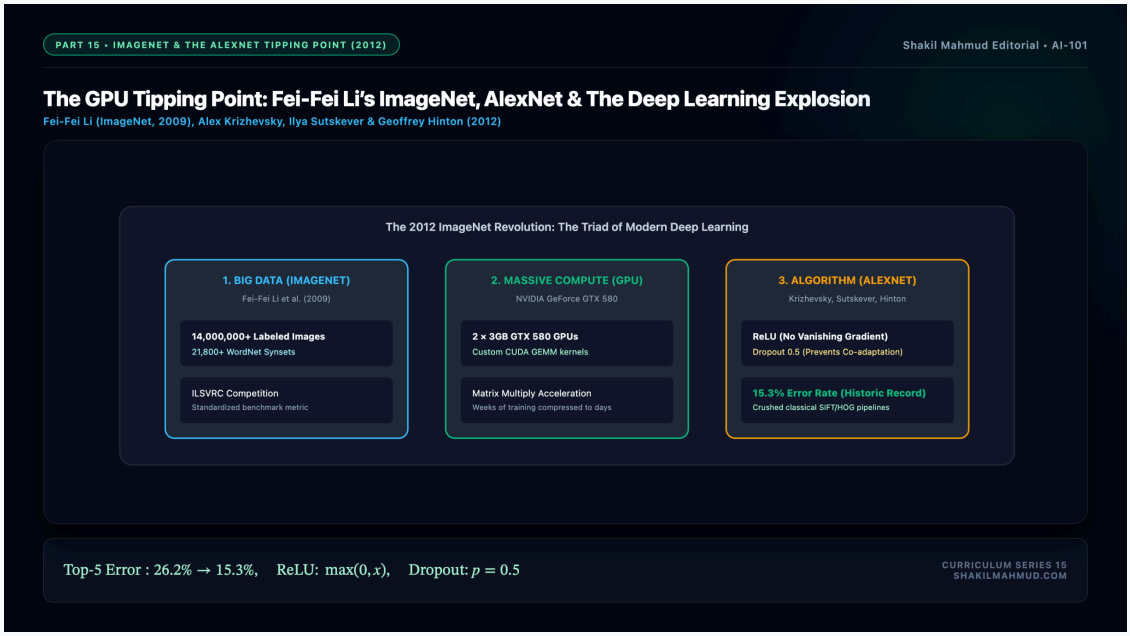


Plate XV: *The 2012 Deep Learning tipping point: Fei-Fei Li's ImageNet dataset, NVIDIA GPU acceleration, and AlexNet's ReLU/Dropout architecture.*

Despite their immense capabilities, Large Language Models are static artifacts: their parametric memory is frozen at training cutoff, and their stochastic generative nature induces **Hallucinations**—convincing fabrications presented with unflinching confidence. **Retrieval-Augmented Generation (RAG)** unites non-parametric external knowledge bases with parametric neural generation, dynamically retrieving verified factual context from petabyte-scale vector indices and grounding model outputs in verifiable evidence.

1. The Tripartite Architecture of RAG

Formulated by Patrick Lewis et al. (2020), modern enterprise RAG decouples knowledge retrieval from reasoning across three distinct stages:

1. **Document Chunking & Vectorization:** Unstructured corpora are partitioned into semantic chunks $dinmathcalD$ and encoded into continuous vector spaces via a dense bi-encoder: $mathbfe_d = E_textdoc(d)inmathbbR^D$.
2. **Approximate Nearest Neighbor (ANN) Retrieval:** Given user query q, encode $mathbfe_q = E_textquery(q)$ and retrieve the top-k most similar documents maximizing cosine similarity: $textsim(q, d) = fracmathbfe_qcdotmathbfe_d|mathbfe_q||mathbfe_d|$.
3. **Conditioned Generation:** The retrieved documents are formatted into an augmented context prompt passed to the generator: P(y mid q) = prod_t P(y_t mid q, d_1, dots, d_k, y_{<t}).

2. Hierarchical Navigable Small World (HNSW) Indexing

Exact nearest neighbor search scales as $O(NcdotD)$, which is intractable for millions of documents. **HNSW** (Malkov & Yashunin, 2018) constructs a multi-layer geometric graph structure inspired by skip-lists:

Shakil Mahmud • Research Monograph

- **Layer Hierarchy:** The top layers contain sparse, long-range connections between distant vector clusters, enabling rapid logarithmic geographic traversal across the vector space.
- **Greedy Graph Traversal:** At each layer, search begins at an entry point and greedily moves to the neighboring node that minimizes Euclidean distance to query vector $mathbfe_q$. Upon reaching a local minimum, search drops to the denser layer below.
- **Bottom Layer ell_0:** The base layer holds all vectors connected via Delaunay-like clustering graphs, achieving $O(logN)$ search latency with >98% recall accuracy.

3. Hybrid Search & Cross-Encoder Reranking

Dense semantic retrieval often fails when users search for exact product codes, legal IDs, or rare acronyms. Modern production systems employ **Hybrid Retrieval**, fusing dense vector similarity with sparse lexical keyword matching (BM25) via **Reciprocal Rank Fusion (RRF)**:

text{RRF}(d) = sum_{m in {text{dense}, text{sparse}}} frac{1}{k + text{Rank}_m(d)}

Candidate pools from hybrid retrieval (e.g., top 100 candidates) are then passed through an intensive **Cross-Encoder Reranker**. Unlike bi-encoders that process queries and documents independently, cross-encoders concatenate the query and document into a single sequence $[q; d]$ and compute full all-to-all attention across all tokens, scoring relevance with precision that eliminates irrelevant noise before LLM ingestion.



Citation: Mahmud, S. (2026). "Retrieval-Augmented Generation (RAG) — Dense Embeddings, HNSW Indexing & Hybrid Reranking." In Large Language Models & Generative AI, Part 15. https://shakilmahmud.com/large-language-models-part-15-retrieval-augmented-generation-rag-dense-embeddings-hnsw/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XVI

Autoregressive Decoding Strategies — Temperature, Top-P Nucleus & Speculative Decoding

Autoregressive Decoding Strategies — Temperature, Top-P Nucleus & Speculative Decoding

454 words • November 4, 2026 • Permanent URI

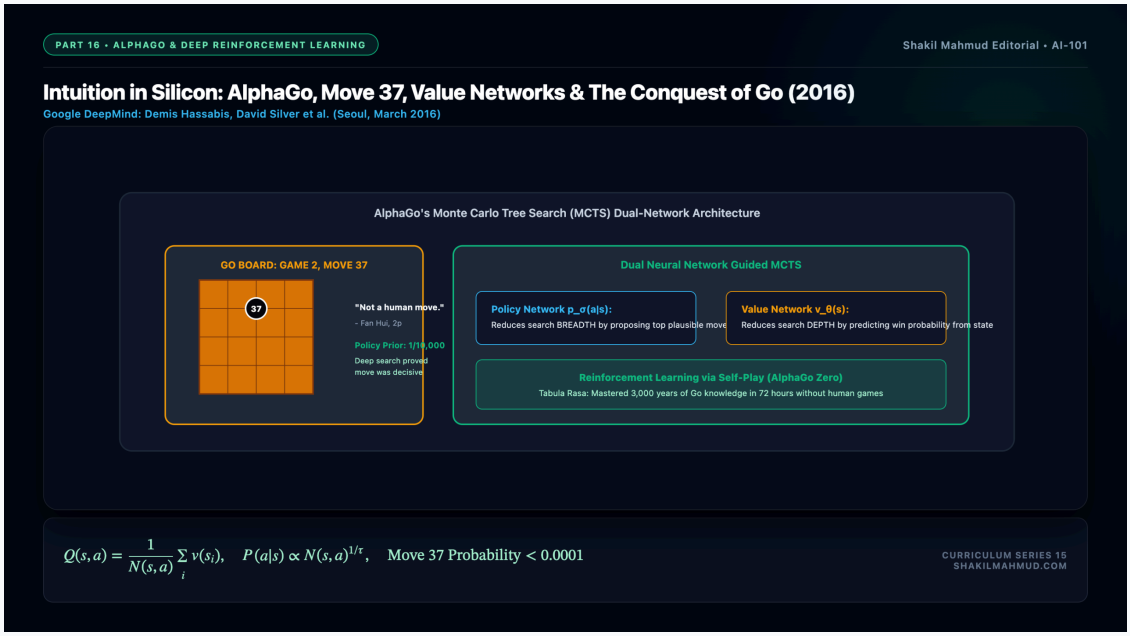


Plate XVI: *AlphaGo's dual policy and value networks guiding Monte Carlo Tree Search, immortalized by the intuitive Move 37 shoulder hit.*

The final forward pass of a Large Language Model does not output words; it produces a vector of unnormalized logits $mathbf{z} in mathbb{R}^{|mathcal{V}|}$ across its vocabulary. The algorithms that convert these continuous logits into discrete, flowing sentences—**Decoding Strategies**—dictate whether the model produces repetitive, rigid text, creative prose, or incoherent gibberish. Furthermore, modern innovations like **Speculative Decoding** fundamentally accelerate the serial generation bottleneck without altering the target distribution.

1. Thermodynamic Logit Temperature Scaling

Logits are converted into probability distributions via Softmax. Introducing temperature parameter T > 0 alters the entropy of the Boltzmann distribution:

P(w_i) = frac{exp(z_i / T)}{sum_{j=1}^{|mathcal{V}|} exp(z_j / T)}

- $T to 0$ **(Greedy Search):** The distribution collapses into a Dirac delta on the maximum logit ($argmax z_i$). This minimizes hallucinations in mathematics and coding, but often causes severe cyclic repetitions in creative dialogue.
- $T = 1.0$**:** Exact sampling from the true model distribution.
- **T > 1.0:** Flattens the distribution toward a uniform distribution, increasing lexical diversity and creativity at the risk of syntactic breakdown.

2. Truncation Strategies: Top-k vs Top-p (Nucleus) Sampling

Pure temperature sampling suffers from the heavy tail of the vocabulary: rare words with tiny probabilities can occasionally be sampled, destroying coherence.

Shakil Mahmud • Research Monograph

- **Top-k Sampling (Fan et al., 2018):** Hard-truncates the distribution to the k most probable tokens, setting all other logits to $-infty$. The limitation: a fixed $k = 50$ is too restrictive when the distribution is flat, yet too permissive when a single token has 99% probability.
- **Top-p / Nucleus Sampling (Holtzman et al., 2019):** Dynamically truncates the vocabulary to the smallest subset $V^{(p)} subsetmathcal V$ whose cumulative probability exceeds threshold $pin(0, 1]$:
  sum_{w in V^{(p)}} P(w) ge p

  The cutoff expands dynamically when uncertainty is high and contracts tightly when the next token is obvious.

3. Speculative Decoding: Breaking the Sequential Barrier

Autoregressive decoding is inherently serial: generating K tokens requires K sequential forward passes through a massive model (e.g., 70B parameters), which is severely memory-bandwidth bound. **Speculative Decoding** (Leviathan et al., 2023; Chen et al., 2023) breaks this sequential dependency.

A lightweight draft model (e.g., 1B parameters) speculatively generates K candidate tokens rapidly: $(hatx_1, dots, hatx_K)$. The large target model then evaluates all K tokens simultaneously in a **single parallel forward pass**. Each candidate token is accepted or rejected based on a rejection sampling criterion that mathematically preserves the exact distribution of the target model:

P(text{accept } hat{x}_i) = minleft(1, frac{P_{text{target}}(hat{x}_i mid dots)}{P_{text{draft}}(hat{x}_i mid dots)}right)

If token j is rejected, the target model resamples the token and discards the remaining speculative tokens. Because large models are compute-underutilized during single-token decoding, verifying K tokens takes virtually the same time as generating one, accelerating overall inference latency by 2x to 3x with **zero loss in mathematical output fidelity**.

Citation: Mahmud, S. (2026). "Autoregressive Decoding Strategies — Temperature, Top-P Nucleus & Speculative Decoding." In Large Language Models & Generative AI, Part 16.
https://shakilmahmud.com/large-language-models-part-16-autoregressive-decoding-temperature-top-p-speculative/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XVII

Score-Based Continuous Generative Diffusion — Stochastic Differential Equations & DDPM

Score-Based Continuous Generative Diffusion — Stochastic Differential Equations & DDPM

434 words • November 4, 2026 • Permanent URI

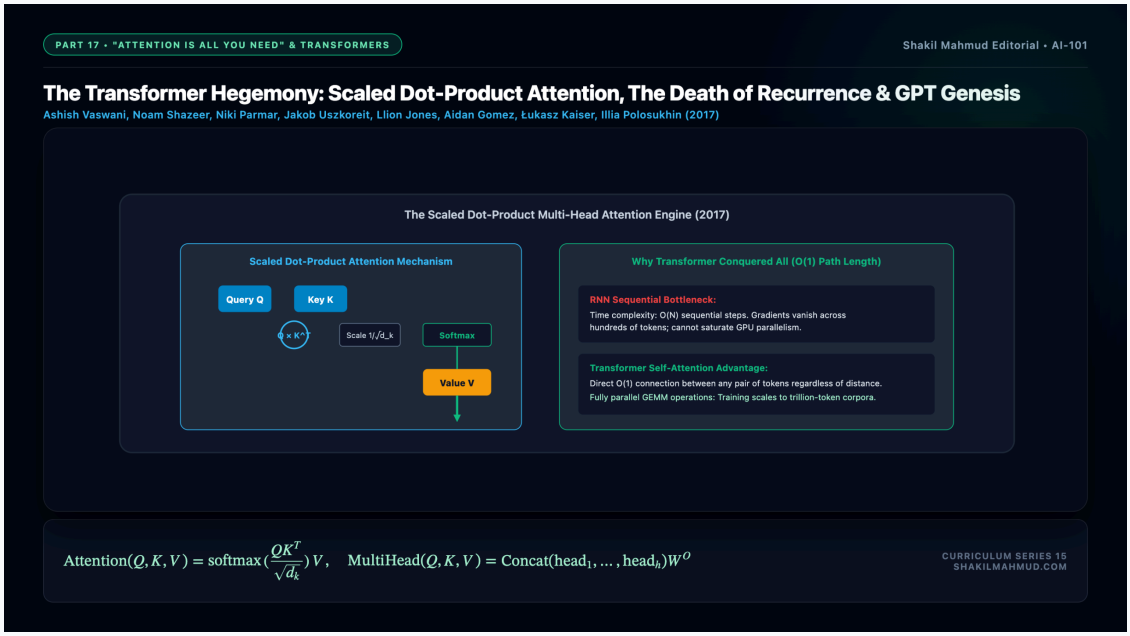


Plate XVII: *The Transformer architecture: Scaled dot-product multi-head self-attention, replacing recurrent bottlenecks with O(1) token interactions.*

While autoregressive Transformers dominate discrete symbolic sequences, continuous high-dimensional generative synthesis—spanning photorealistic visual imagery, video generation, and audio waveform synthesis—is ruled by **Diffusion Models**. Rooted in non-equilibrium statistical thermodynamics, diffusion conceptualizes generative synthesis not as classification, but as an inversion of physical entropy: systematically adding Gaussian noise to an image until information dissolves into pure chaos, and training a neural network to reverse the arrow of time to reconstruct complex structured manifolds.

1. The Forward Noising Process: Ornstein-Uhlenbeck Mechanics

In Denoising Diffusion Probabilistic Models (DDPM, Ho et al., 2020), given a clean data sample $mathbfx_0simq(mathbfx_0)$, a forward Markov chain progressively injects Gaussian noise across T discrete timesteps according to a predefined variance schedule $beta_1, dots, beta_T$:

q(mathbf{x}_t mid mathbf{x}_{t-1}) = mathcal{N}left(mathbf{x}_t; sqrt{1 – beta_t} mathbf{x}_{t-1}, beta_t mathbf{I}right)

Crucially, using the reparameterization trick with alpha_t = 1 – beta_t and cumulative product $baralpha_t = prod_{s=1}^{t} alpha_s$, any arbitrary noisy state $mathbfx_t$ can be sampled in closed form directly from $mathbfx_0$ without computing intermediate steps:

mathbf{x}_t = sqrt{bar{alpha}_t} mathbf{x}_0 + sqrt{1 – bar{alpha}_t} boldsymbol{epsilon}, quad boldsymbol{epsilon} sim mathcal{N}(mathbf{0}, mathbf{I})

As $ttoT$, $baralpha_tto0$, causing $q(mathbfx_T)approxmathcalN(mathbf0, mathbfI)$, transforming any structured data distribution into isotropic Gaussian noise.

Shakil Mahmud • Research Monograph

2. The Reverse Generative Trajectory & Score Matching

Generative synthesis requires inverting this process: sampling $mathbfx_T simmathcalN(mathbf0, mathbfI)$ and traversing backward along the reverse conditional distribution $p_theta(mathbfx_t-1midmathbfx_t)$. While the exact true reverse distribution $q(mathbfx_t-1midmathbfx_t)$ is mathematically intractable, for infinitesimal steps it is provably Gaussian:

p_theta(mathbf{x}_{t-1} mid mathbf{x}_t) = mathcal{N}left(mathbf{x}_{t-1}; boldsymbol{mu}_theta(mathbf{x}_t, t), boldsymbol{Sigma}_theta(mathbf{x}_t, t)right)

Yang Song and Stefano Ermon (2019) demonstrated that reversing this process is equivalent to learning the **Stein Score Function**: the gradient of the log data density with respect to the input space, $nabla_mathbfx log q_t(mathbfx)$.

3. Continuous-Time Formulation via Stochastic Differential Equations (SDEs)

Song et al. (2020) generalized discrete diffusion into a continuous time continuum $tin[0, 1]$ governed by an Itô Stochastic Differential Equation:

mathrm{d}mathbf{x} = mathbf{f}(mathbf{x}, t) mathrm{d}t + g(t) mathrm{d}mathbf{w}

Where $mathbff(mathbfx, t)$ is the drift coefficient, $g(t)$ is the diffusion coefficient, and $mathbfw$ is standard Brownian motion. By Anderson's reverse-time theorem (1982), this forward SDE has a unique, deterministic reverse-time SDE:

mathrm{d}mathbf{x} = left[mathbf{f}(mathbf{x}, t) − g(t)^2 nabla_{mathbf{x}} log p_t(mathbf{x}) right] mathrm{d}t + g(t) mathrm{d}bar{mathbf{w}}

A neural network $boldsymbolepsilon_theta(mathbfx_t, t)$ is trained via Denoising Score Matching to predict the added noise vector:

mathcal{L}_{text{simple}}(theta) = mathbb{E}_{t, mathbf{x}_0, boldsymbol{epsilon}} left[| boldsymbol{epsilon} − boldsymbol{epsilon}_theta(mathbf{x}_t, t) |_2^2 right]

Once trained, the reverse SDE can be integrated using standard numerical ODE solvers (Euler-Maruyama, Runge-Kutta, DDIM), generating breathtaking continuous manifolds from pure stochastic noise.

Citation: Mahmud, S. (2026). "Score-Based Continuous Generative Diffusion — Stochastic Differential Equations & DDPM." In Large Language Models & Generative AI, Part 17.
https://shakilmahmud.com/large-language-models-part-17-score-based-continuous-generative-diffusion-sde-ddpm/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XVIII

Latent Diffusion & Flow Matching — Stable Diffusion, Perceptual Compression & Rectified Flows

Latent Diffusion & Flow Matching — Stable Diffusion, Perceptual Compression & Rectified Flows

448 words • November 4, 2026 • Permanent URI

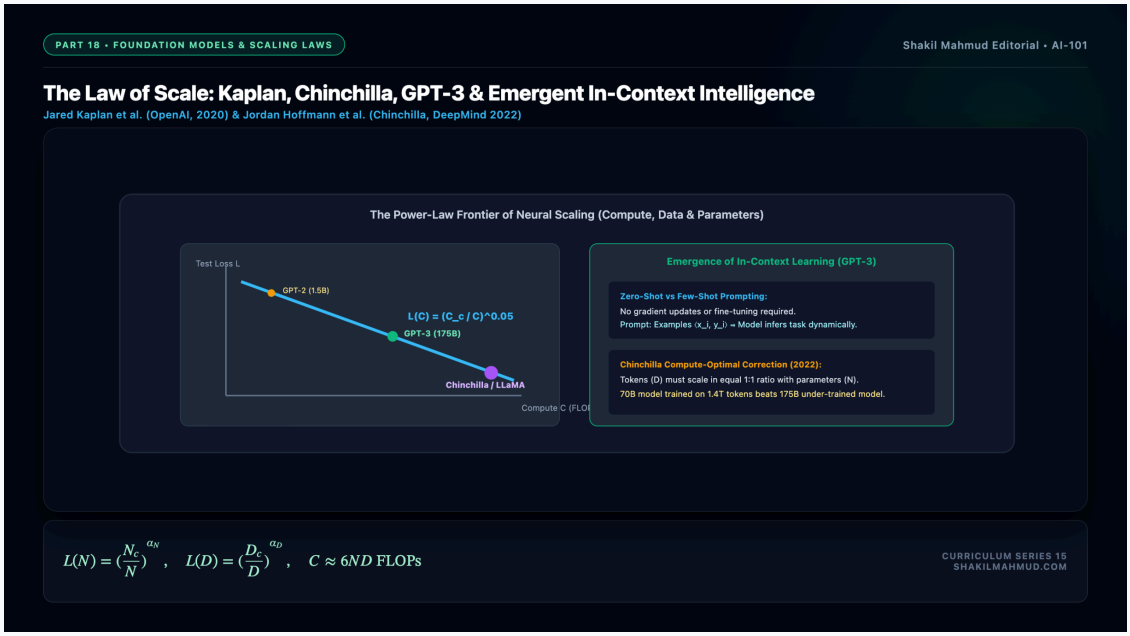


Plate XVIII: *Empirical neural scaling laws: Kaplan & Chinchilla power-law frontiers and the emergence of few-shot in-context learning in GPT-3.*

Executing continuous diffusion directly in high-resolution pixel space ($1024 times 1024 times 3 approx 3 times 10^6$ dimensions) is computationally prohibitive: the network wastes over 90% of its capacity modeling imperceptible high-frequency pixel noise. **Latent Diffusion Models (LDM / Stable Diffusion)** and **Flow Matching** solve this by partitioning generation into perceptual compression and semantic synthesis, propelling generative AI into consumer adoption and millisecond-level sampling.

1. Two-Stage Perceptual Space Decomposition

Rombach et al. (2022) established that visual generative modeling involves two distinct regimes:

1. **Perceptual Compression:** An autoencoder (VAE) trained with adversarial and perceptual losses compresses an RGB image $mathbf x in mathbb R^{H times W times 3}$ into a low-dimensional continuous latent space $mathbf z = mathcal E(mathbf x) in mathbb R^{h times w times c}$, typically with downsampling factor $f = H/h = 8$. Spatial resolution is reduced 64-fold, filtering out imperceptible noise while preserving semantic geometry.
2. **Semantic Synthesis:** The diffusion backbone (U-Net or Diffusion Transformer / DiT) operates exclusively within the compact latent space $mathbf z$, slashing FLOPs by orders of magnitude. The final latent is decoded into full resolution via $hat mathbf x = mathcal D(mathbf z)$.

2. Conditioning & Classifier-Free Guidance (CFG)

To steer image generation with textual prompts y, text is embedded via a pre-trained frozen encoder (CLIP ViT-L or T5-XXL) into embeddings $mathbf c = tau_ theta(y)$ and injected into diffusion layers via cross-attention:

Shakil Mahmud • Research Monograph

text{Attention}(mathbf{Q}, mathbf{K}, mathbf{V}) = text{Softmax}left(frac{mathbf{Q} mathbf{K}^T}{sqrt{d}}right) mathbf{V}, quad mathbf{Q} = mathbf{W}_Q mathbf{z}_t, quad mathbf{K} = mathbf{W}_K mathbf{c}, quad mathbf{V} = mathbf{W}_V mathbf{c}

To guarantee strict adherence to the prompt, Jonathan Ho and Tim Salimans (2021) formulated **Classifier-Free Guidance (CFG)**. During inference, the predicted noise is computed as an extrapolation between the conditioned prediction and an unconditioned prediction (passing empty prompt $emptyset$):

tilde{boldsymbol{epsilon}}_theta(mathbf{z}_t, mathbf{c}) = boldsymbol{epsilon}_theta(mathbf{z}_t, emptyset) + s cdot left(boldsymbol{epsilon}_theta(mathbf{z}_t, mathbf{c}) – boldsymbol{epsilon}_theta(mathbf{z}_t, emptyset) right)

Guidance scale s > 1 (typically $approx 7.5$) amplifies text alignment and color saturation while pruning improbable visual artifacts.

3. The Flow Matching Paradigm: Rectified Straight Trajectories

Standard diffusion trajectories are curved Brownian paths, requiring 30 to 50 solver steps to integrate. **Flow Matching** (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2022) reformulates generative modeling as learning a vector field $v_t(mathbf{x})$ that generates a Continuous Normalizing Flow:

frac{mathrm{d}mathbf{x}_t}{mathrm{d}t} = v_t(mathbf{x}_t)

Under **Rectified Flow**, the probability paths between pure noise $mathbf{x}_0$ and real data $mathbf{x}_1$ are defined as straight lines:

mathbf{x}_t = (1 – t) mathbf{x}_0 + t mathbf{x}_1, quad frac{mathrm{d}mathbf{x}_t}{mathrm{d}t} = mathbf{x}_1 – mathbf{x}_0

The neural network simply minimizes the regression objective: mathcal{L}_{text{FM}}(theta) = mathbb{E} [| v_theta(mathbf{x}_t, t) – (mathbf{x}_1 – mathbf{x}_0) |²]. Because the trajectories are straight lines, the ODE can be integrated in as few as **1 to 4 steps**, powering modern frontier engines like Stable Diffusion 3 and Flux.

Citation: Mahmud, S. (2026). "Latent Diffusion & Flow Matching — Stable Diffusion, Perceptual Compression & Rectified Flows." In Large Language Models & Generative AI, Part 18. https://shakilmahmud.com/large-language-models-part-18-latent-diffusion-flow-matching-stable-diffusion-rectified/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XIX

Long-Context Systems — RingAttention, Blockwise Context & Needle-In-A-Haystack Limits

Long-Context Systems — RingAttention, Blockwise Context & Needle-In-A-Haystack Limits

505 words • November 4, 2026 • Permanent URI

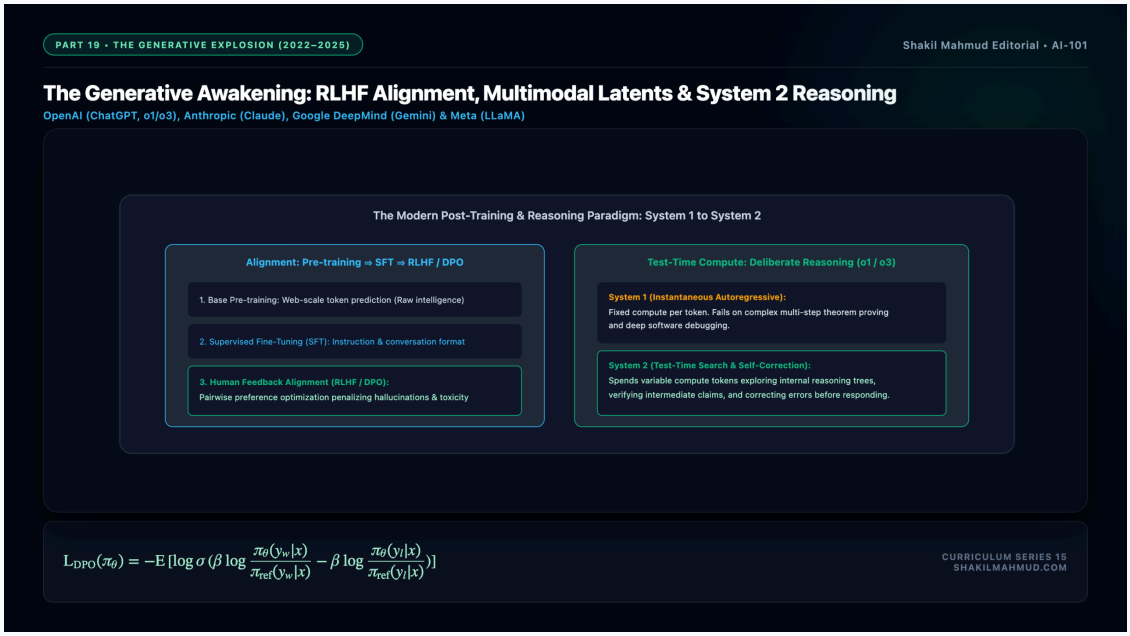


Plate XIX: *The generative AI paradigm: From RLHF/DPO preference alignment to System 2 test-time reasoning search in modern frontier models.*

Early foundation models were constrained to context windows of 2,048 tokens—scarcely enough for a short essay. Today, state-of-the-art architectures routinely ingest 1 to 2 million tokens in a single context window, encompassing whole codebases, legal archives, and hours of video. Scaling context lengths across orders of magnitude required overcoming both the algebraic quadratic memory explosion of attention and the positional extrapolation failure of harmonic embeddings via **RingAttention** and **RoPE Frequency Scaling**.

1. Positional Extrapolation & RoPE Frequency Adjustments

In Rotary Position Embedding (RoPE), positional information is injected by rotating query and key vectors in complex 2D planes via frequencies $theta_i = b^{-2(i-1)/d}$, where base $b = 10,000$. When a model trained on $L = 4,096$ tokens is evaluated on $L = 32,768$ tokens, it encounters unseen relative rotation angles, causing attention scores to blow up catastrophically.

To expand context without retraining from scratch, several frequency interpolation methods were pioneered:

- **Linear Position Interpolation (PI, Chen et al., 2023):** Linearly compresses position indices by scale factor $alpha = L_{textnew}/L_{textold}$: m’ = m / alpha. This preserves relative angles within the original training envelope.
- **NTK-Aware RoPE Scaling (bloc97, 2023):** Rather than scaling all frequencies uniformly, NTK-aware scaling adjusts the base frequency b: b’ = b cdot alpha^{frac{d}{d – 2}}

  This preserves high-frequency components that capture critical local syntax while compressing low-frequency components that encode macroscopic sequence positions.
- **YaRN (Peng et al., 2023):** Applies a ramped weighting function between high and low frequencies combined with temperature scaling to prevent attention entropy degradation at million-token horizons.

Shakil Mahmud • Research Monograph

2. RingAttention: Distributed Ring Communication Topology

Even with linear attention approximations, storing the KV cache for 1 million tokens across 32 layers exceeds the memory capacity of a single GPU (consuming >100 GB). **RingAttention** (Hao Liu, Matei Zaharia, & Pieter Abbeel, 2023) distributes long sequences across an entire cluster of P GPUs arranged in a logical ring network.

1. Each GPU $pin0, dots, P-1$ holds a local slice of queries $mathbfQ^{(p)}$ and keys/values $mathbfK^{(p)}, mathbfV^{(p)}$ of size N/P.
2. In parallel with computing local attention between $mathbfQ^{(p)}$ and the current local $mathbfK, mathbfV$ block in fast SRAM, GPU p asynchronously sends its $mathbfK, mathbfV$ block to its neighbor $p+1$ via high-speed NVLink/P2P while receiving the next block from neighbor $p-1$.
3. Using online softmax accumulation, the local attention accumulator is updated step-by-step.
4. After P ring shifts, each GPU has attended to the entire global context window.

Communication perfectly overlaps with computation, eliminating memory limits and scaling sequence length linearly with cluster size up to millions of tokens.

3. The Needle-In-A-Haystack (NIAH) Benchmark & Real Limits

To verify whether a model truly digests its vast context rather than merely ignoring it, Greg Kamradt formulated the **Needle In A Haystack (NIAH)** evaluation: a single arbitrary fact (e.g., *"The secret password for the vault is 84920"*) is embedded at random depth percentages (0% to 100%) inside vast distraction corpora up to 1,000,000 tokens. Modern frontier models achieve 100% retrieval fidelity across the entire grid, proving that long-context attention behaves as a near-perfect associative memory bank.



Citation: Mahmud, S. (2026). "Long-Context Systems — RingAttention, Blockwise Context & Needle-In-A-Haystack Limits." In Large Language Models & Generative AI, Part 19.
https://shakilmahmud.com/large-language-models-part-19-long-context-systems-ringattention-rope-needle-haystack/

SHAKIL MAHMUD RESEARCH MONOGRAPHS • PART XX

The Frontiers of Synthetic Cognition — Multimodality, Reasoning Tokens & Test-Time Compute

The Frontiers of Synthetic Cognition — Multimodality, Reasoning Tokens & Test-Time Compute

404 words • November 4, 2026 • Permanent URI

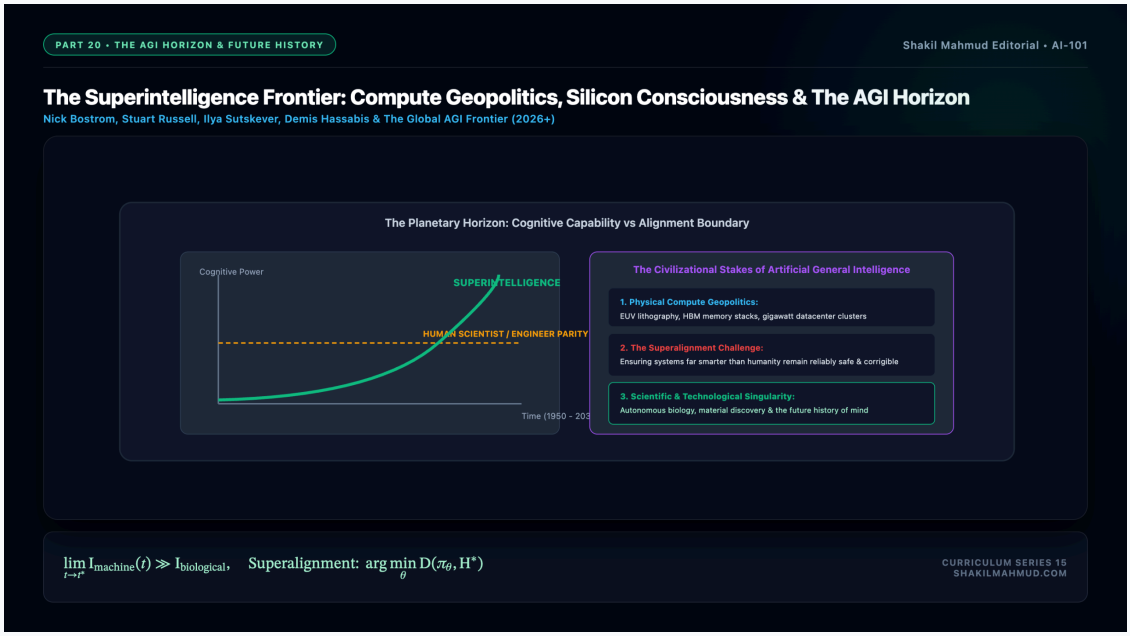


Plate XX: *The Artificial General Intelligence frontier: Exponential cognitive capability curves, EUV compute infrastructure, and superalignment theory.*

The grand narrative of generative AI is culminating in a historic convergence: the boundary between text, vision, code, and structured reasoning has dissolved. Modern frontier models are no longer purely linguistic mimics; they are becoming unified, multi-modal cognitive engines. By trading inference-time compute for algorithmic deliberation through **Reasoning Tokens** and **Test-Time Search**, generative systems are breaking the pre-training scaling plateau and pioneering an entirely new dimension of artificial cognition.

1. Unified Multimodal Latent Spaces

Modern multimodal foundation models (GPT-4o, Gemini 1.5, Claude 3.5 Sonnet) reject separate external captioning modules in favor of native multi-modal tokenization:

- **Vision Tokenization:** Continuous visual patches from vision transformers (e.g., SigLIP, ViT-22B) are projected directly into the LLM embedding manifold via lightweight cross-attention projectors or MLP adapters: $mathbfe_v = mathbfW_2cdottextGELU(mathbfW_1mathbfv)$.
- **Any-to-Any Modality:** Audio waveforms, video streams, and tactile robotics signals are tokenized into identical discrete or continuous token vectors. Self-attention layers compute cross-modal relationships natively, allowing a single neural network to reason fluidly across sound, sight, and symbol.

2. Test-Time Compute: The New Scaling Frontier

For seven years, AI capabilities scaled primarily along two dimensions defined by Kaplan and Chinchilla: *Pre-training Compute* and *Dataset Size*. However, as high-quality human text is exhausted and physical datacenter power limits loom, a third scaling law has emerged: **Inference-Time (Test-Time) Compute Scaling** (OpenAI o1/o3, 2024).

Shakil Mahmud • Research Monograph

Given a challenging Olympiad-level mathematical or programming problem, allocating more compute during inference allows the model to search through reasoning trees, explore counterfactual hypotheses, backtrack from dead ends, and verify its own deductions:

text{Performance} propto f(text{Test-Time Compute FLOPs})

Rather than outputting an answer immediately, the model emits thousands of hidden **Reasoning Tokens** $langletextthoughtrangledotslangle/textthoughtrangle$ trained via large-scale reinforcement learning with verifiable process reward models. A smaller model with extensive test-time deliberation routinely outperforms models 10x its size executing a single greedy forward pass.

3. Autonomous Agentic Loops & The Horizon of Generative Synthesis

Generative models are evolving from passive question-answering engines into active, autonomous agents capable of interacting with the physical and digital world:

- **Tool Execution & Environment Feedback:** Generating executable code, invoking APIs, compiling environments, and incorporating execution errors back into context to self-correct.
- **Synthetic Data Flywheels:** Frontier models generating high-quality curricula to train successor generations, initiating recursive self-improvement loops.
- **The Philosophical Threshold:** As generative systems master formal mathematics, scientific discovery, and creative literature, humanity stands at the threshold of synthetic intelligence—a transition from language modeling to the creation of autonomous artificial reason.

Citation: Mahmud, S. (2026). "The Frontiers of Synthetic Cognition — Multimodality, Reasoning Tokens & Test-Time Compute." In Large Language Models & Generative AI, Part 20.
https://shakilmahmud.com/large-language-models-part-20-frontiers-synthetic-cognition-multimodal-reasoning-test-time/

PRIMARY SOURCES & SCHOLARLY REFERENCES

Consolidated Bibliography

❖

The formulations, historical contexts, and conceptual developments presented throughout this monograph draw upon the following primary treatises, foundational papers, and critical editions:

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). *Attention Is All You Need*. NeurIPS 30: 5998–6008.

Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). *Scaling Laws for Neural Language Models*. arXiv:2001.08361 [cs.LG].

Brown, T. B., Mann, B., Ryder, N., et al. (2020). *Language Models are Few-Shot Learners*. NeurIPS 33: 1877–1901.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. NeurIPS 35: 16344–16359.

Hu, E. J., Shen, Y., Wallis, P., et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. ICLR 2022.

Rafailov, R., Sharma, A., Mitchell, E., et al. (2023). *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. NeurIPS 36.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., et al. (2020). *Score-Based Generative Modeling through Stochastic Differential Equations*. ICLR 2021.

Shakil Mahmud • Research Monograph

BIOGRAPHICAL NOTE

About the Author

❖

Shakil Mahmud

Independent Scholar, Writer & Systems Architect

Shakil Mahmud is an independent scholar, author, and software systems architect. His research interests span the foundations of theoretical physics, the history and philosophy of mathematics, evolutionary biology, macroeconomic history, and the architecture of neural computing. His work synthesizes deep historical rigor with accessible first-principles exposition. His complete published archives and monographs are maintained at **shakilmahmud.com**.

COLOPHON

Typeset in EB Garamond (English) and Kohinoor Bangla / Noto Serif Bengali (Bengali).
Mathematical typesetting generated with Donald Knuth's TeX algorithms and KaTeX.
Published digitally as a monolithic research monograph by Shakil Mahmud Editorial Series.
Dhaka & Singapore • 2026 • https://shakilmahmud.com