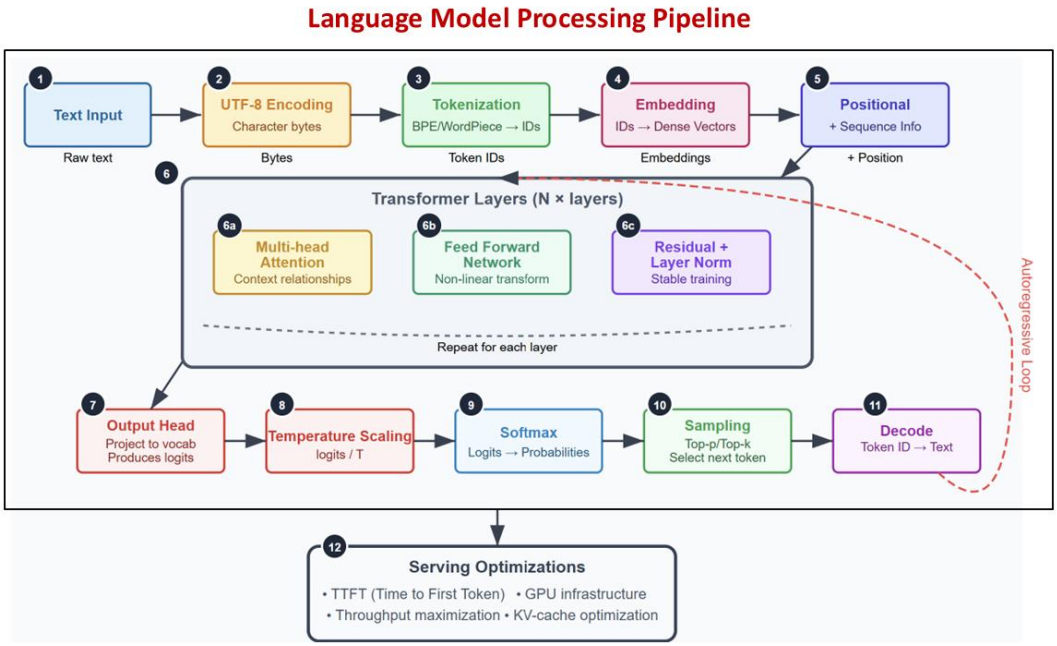
GenAI Glossary - Binary to Deployment

A complete glossary covering core concepts from binary representation to production deployment.

---

26. Language Model Processing Flow: Text → Tokens → Model → Output



L1 - Foundations

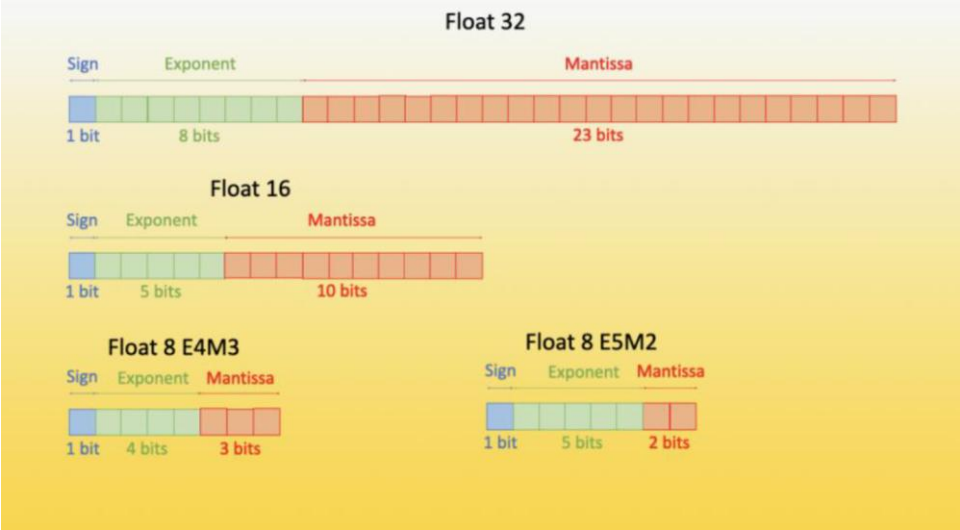
1. Binary & Data Foundations

Core Data Units

- **Bit** → Smallest unit of information (0 or 1). Hardware foundation for all computing.
- **Byte** → 8 bits grouped together. Can represent 256 values (0-255). Basic storage unit.
- **Word (hardware)** → CPU's natural data unit size (32-bit, 64-bit). Important for memory alignment and processing efficiency.

Number Representations

- **Integer (INT)** → Whole numbers in binary with fixed bit length (INT4, INT8, INT16, INT32, INT64). Used in quantized models. Also, INT1 (binary) for extreme compression in some specialized models
- **Floating Point** → Real numbers stored in scientific notation format:
    - **FP32** → Full precision 32-bit float. Standard for training.
    - **FP16** → Half precision 16-bit float. Saves memory but less precise.
    - **FP8** → Ultra-low precision format becoming important for edge deployment
    - **E4M3** and **E5M2** formats → emerging 8-bit floating point standards gaining traction for inference
    - **BF16** → Brain Float 16-bit format with larger dynamic range than FP16, avoids underflow issues.



Source: https://huggingface.co/blog/4bit-transformers-bitsandbytes

- **Quantization** → Technique reducing precision (e.g., FP32 → INT8/4-bit) to make models smaller and faster with minimal accuracy loss.

---

2. Text Representation & Encoding

Character Systems

- **Character** → Human-readable symbol (a, 中, @). Not fixed in byte size.
- **Unicode** → Universal standard assigning unique numeric code points to every character in every language (e.g., U+0041 = A).

Encoding Schemes

- **ASCII** → 7-bit encoding for English letters, digits, punctuation (values 0-127).
- **Extended ASCII** → 8-bit variant adding 128 extra symbols (0-255). Limited for international text.
- **UTF-8** → Variable-length encoding (1-4 bytes per character). Supports all Unicode. Global web standard.
- **UTF-16/UTF-32** → Fixed-length alternatives (2 or 4 bytes per char). Used internally in some programming languages.

---

3. Tokenization (Text → Model Input)

Core Concepts

- **Token** → Basic unit LLMs process. Could be a word (apple), subword (ap, ##ple), or punctuation (!).
- **Subword** → Partial word token. Handles rare/unknown words by splitting into smaller pieces.
- **Token ID** → Integer index assigned to each token in the vocabulary (actual model input).
- **Vocabulary** → Fixed set of all tokens a model knows (e.g., GPT-3 ~50k tokens, GPT-4 ~100k).

Tokenization Algorithms

Byte Pair Encoding (BPE)

- **Process:** Starts with character-level vocabulary, iteratively identifies most frequent adjacent pairs and merges them into new tokens
- **Algorithm:** Counts all symbol pairs → merges most frequent → repeats until desired vocab size

- **Advantages:** Handles rare/OOV words well, compact representation
- **Disadvantages:** Can create suboptimal segmentations, language-dependent preprocessing needed
- **Used in:** GPT-2, GPT-3, RoBERTa
- **Example:** "tokenization" → "token" + "ization" → "token" + "iz" + "ation"

WordPiece

- **Process:** Similar to BPE but uses likelihood-based merging rather than frequency
- **Key Difference:** Selects merges that maximize training data likelihood, not just frequency
- **Prefix Convention:** Uses "##" to indicate subword continuations
- **Advantages:** More principled than BPE, better handling of morphology
- **Used in:** BERT, DistilBERT, ELECTRA
- **Example: "**tokenization" → "token" + "##ization"

SentencePiece

- **Key Innovation:** Treats whitespace as normal characters, no pre-tokenization needed
- **Language Agnostic:** Works directly on raw text without language-specific rules
- **Supports:** Both BPE and unigram language model algorithms underneath
- **Advantages:** Unified handling of all languages, reversible tokenization
- **Used in:** T5, mT5, XLNet, many multilingual models
- **Example:** Handles "Hello world" and "こんにちは" with same algorithm

Byte-level Tokenization

- **Core Concept**: Every input mapped to UTF-8 bytes, guaranteeing finite vocabulary
- **Vocabulary Size**: Base vocab of 256 bytes + learned merges
- **Advantages**: No unknown tokens ever, handles any text input, multilingual by design
- **Trade-offs**: Longer sequences for non-ASCII text, more tokens for same content
- **Used in**: GPT-4, Claude, Llama models
- **Example**: Emoji, special symbols, any Unicode character can be represented

---

4. Information Theory & Measurement

Compression & Entropy

- **Entropy** → Measures unpredictability/information content of data. Higher entropy = more random/harder to predict.
- **Bits per Character (BPC)** → Average bits required to encode one character. Lower BPC = better compression/prediction.
- **Bits per Byte (BPB)** → Similar to BPC but computed per byte. Useful in compression research.

Loss & Evaluation Metrics

- **Cross-Entropy** → Measures distance between predicted and true probability distributions. Primary training loss.
- **Perplexity** → 2^(cross-entropy). Interpretable measure of model "confusion." Lower = better predictions.
- **Log-Likelihood** → Sum of log probabilities for correct tokens. Used for evaluation and training objectives.
- **Negative Log-Likelihood (NLL)** → Equivalent to cross-entropy loss. Penalizes low probabilities on correct tokens.

---

5. Probability Concepts in LLMs

Core Probability

- **Probability Distribution** → LLMs predict probability of each token being next. Must sum to 1.0.
- **Conditional Probability** → P(token | previous tokens). Core of autoregressive LLMs.
- **Joint Probability** → Probability of entire sequence = product of conditional probabilities.
- **Softmax Function** → Converts raw model scores (logits) into valid probabilities that sum to 1.

Sampling Strategies

- **Greedy Sampling** → Always pick highest probability token. Fast but repetitive.
- **Random Sampling** → Sample according to exact probability distribution.
- **Top-k Sampling** → Restrict choices to top k highest-probability tokens, then sample.
- **Top-p (Nucleus) Sampling** → Choose smallest set of tokens whose cumulative probability ≥ p, then sample.

- **Temperature Scaling** → Adjust randomness by scaling logits before softmax. Low = deterministic, high = diverse.

Advanced Probability

- **KL Divergence** → Measures distance between two probability distributions. Used in fine-tuning, distillation, RLHF.
- **Beam Search** → Maintains multiple high-probability sequences in parallel. Trades diversity vs. optimality.
- **Maximum Likelihood Estimation (MLE)** → Core training principle: maximize probability of observed training sequences.

---

6. Statistical Concepts

Descriptive Statistics

- **Mean/Average** → Used in embedding normalization, attention scaling, layer statistics.
- **Variance/Standard Deviation** → Critical for layer normalization, weight initialization (Xavier, Kaiming).
- **Min/Max/Range** → Used for gradient clipping, logit clipping, activation bounds.
- **Percentiles/Quantiles** → Applied in quantization strategies and model pruning.

Probability Distributions

- **Normal (Gaussian)** → Standard for weight initialization, noise injection, uncertainty modeling.
- **Uniform Distribution** → Alternative initialization strategy, random sampling baselines.
- **Multinomial** → Models token sampling from softmax outputs.
- **Zipf's Law** → Natural language follows heavy tailed word frequency distribution.

Statistical Methods

- **Maximum A Posteriori (MAP)** → Bayesian extension of MLE incorporating priors.
- **Bias-Variance Tradeoff** → Model complexity vs. generalization ability.
- **Central Limit Theorem** → Why averaged embeddings/weights tend toward Gaussian.
- **Monte Carlo Methods** → Approximate complex expectations in RLHF, uncertainty quantification.

Distance & Similarity Metrics

- **Cosine Similarity** → Comparing embeddings for semantic similarity, retrieval systems.
- **Euclidean Distance** → L2 distance for embedding space geometry.
- **Jensen-Shannon Divergence** → Symmetric version of KL divergence, used in evaluation.

---

7. Traditional ML & Neural Network Foundations

Pre-Transformer Architectures

- **Recurrent Neural Networks (RNNs)** → Sequential neural networks maintaining hidden states to capture temporal dependencies in data.
- **Long Short-Term Memory (LSTM)** → Specialized RNN variant addressing vanishing gradients with memory cells and gating mechanisms.
- **Gated Recurrent Unit (GRU)** → Simplified LSTM variant with fewer parameters and similar performance.

Alternative Generative Models

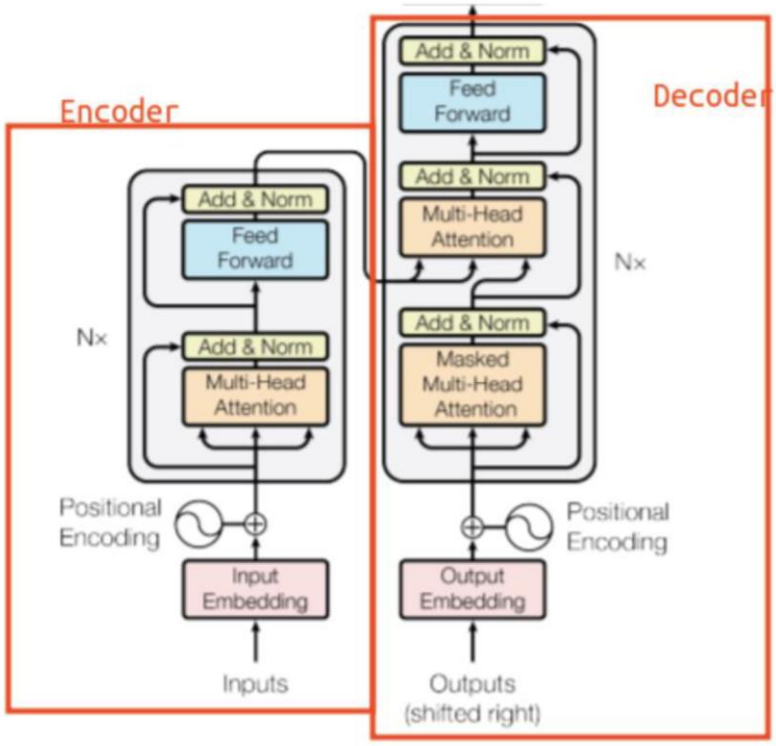
- **Generative Adversarial Networks (GANs)** → Framework with generator and discriminator networks competing to generate realistic data.
- **Variational Autoencoders (VAEs)** → Generative models learning latent representations and sampling from latent space to generate new data.
- **Diffusion Models** → Start from noise and progressively generate structured data through reverse diffusion process.

Fundamental Training Concepts

- **Backpropagation** → Algorithm for updating neural network weights by propagating error gradients backward through layers.
- **Dropout** → Regularization technique randomly ignoring neurons during training to prevent overfitting.
- **Parameter Sharing** → Reusing parameters across different parts of model to reduce total parameters and improve generalization.

L2 - Models & Training

8. Model Architectures & Variants



Transformer encoder-decoder model diagram (Attention is all you need)

Source: https://vaclavkosar.com/ml/Encoder-only-Decoder-only-vs-Encoder-Decoder-Transfomer

Architecture Types

- **Encoder-Only → BERT-style models**: Focused on understanding tasks (classification, sentiment analysis, NER). They learn bidirectional context, making them strong for extracting meaning rather than generating text.
- **Decoder-Only → GPT-style models**: Autoregressive in nature, predicting the next token step by step. Best suited for text generation, completion, and conversational tasks.
- **Encoder-Decoder → T5-style models**: Use an encoder to process input into representations and a decoder to generate outputs. Effective for sequence-to-sequence tasks like translation, summarization, and question answering.
- **Mixture of Experts (MoE) → Sparse models**: Contain many expert sub-networks but activate only a small subset per token. This allows scaling model size without linearly increasing compute cost, making them efficient for very large LLMs.

Attention Variants

- **Cross-Attention** → Attending between different sequences (encoder-decoder).
- **Self-Attention** → Tokens attending to other tokens in same sequence.
- **Causal Attention** → Masked attention preventing future token access.
- **Bidirectional Attention** → Can attend to both past and future tokens.

Scale Categories

- **Small Models** → <1B parameters (mobile, edge deployment).
- **Medium Models** → 1B-10B parameters (efficient serving).
- **Large Models** → 10B-100B parameters (high capability).
- **Foundation Models** → 100B+ parameters (GPT-4x, Gemini, Claude).

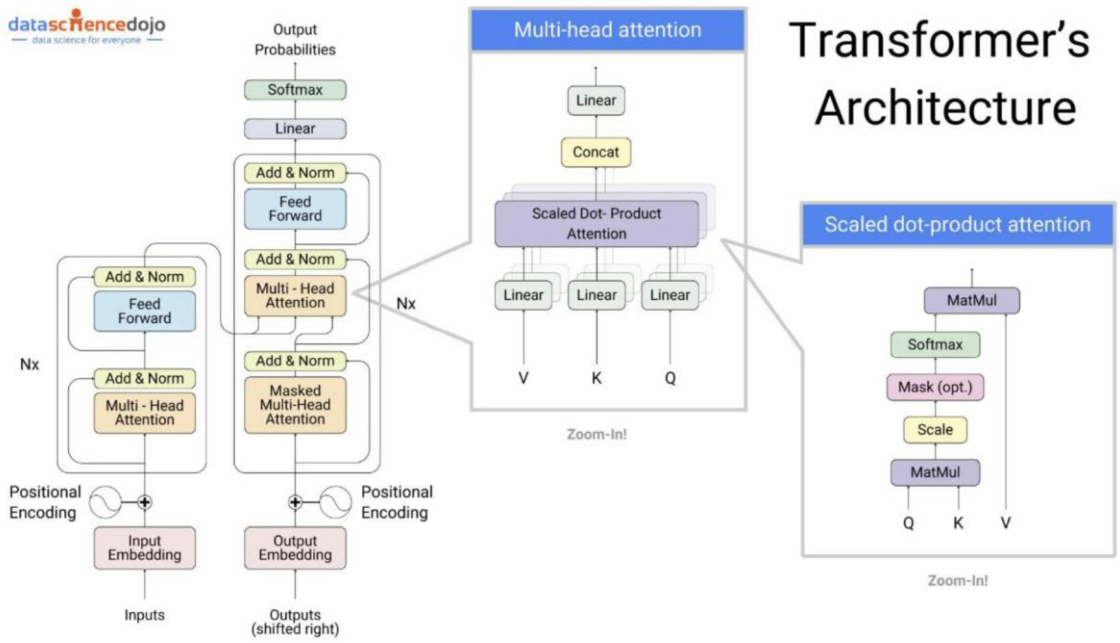
---

9. Model Internals

Input Representation

- **Embedding** → Dense continuous vector representation of tokens (e.g., 768, 1024, 4096 dimensions).
- **Positional Encoding** → Adds sequence order information since transformers lack built-in order:
    - **Absolute** → Fixed sinusoidal position vectors.
    - **RoPE (Rotary)** → Encodes positions using rotations in vector space. More flexible.
- **Context Window** → Maximum tokens model can process simultaneously (e.g., GPT-4: 32k-128k tokens).
- **Attention Mask** → Binary matrix controlling which tokens can "attend" to others.

Transformer Components



The general architecture of transformer models

Source: https://datasciencedojo.com/blog/transformer-models-types-their-uses/

- **Attention Mechanism** → Core innovation computing how much each token should focus on others:
    - **Q (Query)** → Representation of current token asking questions.
    - **K (Key)** → Representation of candidate tokens being queried.
    - **V (Value)** → Information content carried by candidate tokens.
    - **Attention Score** → Dot product of Q and K, scaled by √(dimension).
- **Multi-Head Attention** → Multiple parallel attention mechanisms capturing different linguistic relations.
- **Feed-Forward Network (FFN)** → Multi-layer perceptron applying nonlinear transformations after attention.
- **Residual Connections** → Skip connections adding input back to output, preventing vanishing gradients.
- **Layer Normalization** → Normalizes activations across features per token for training stability.

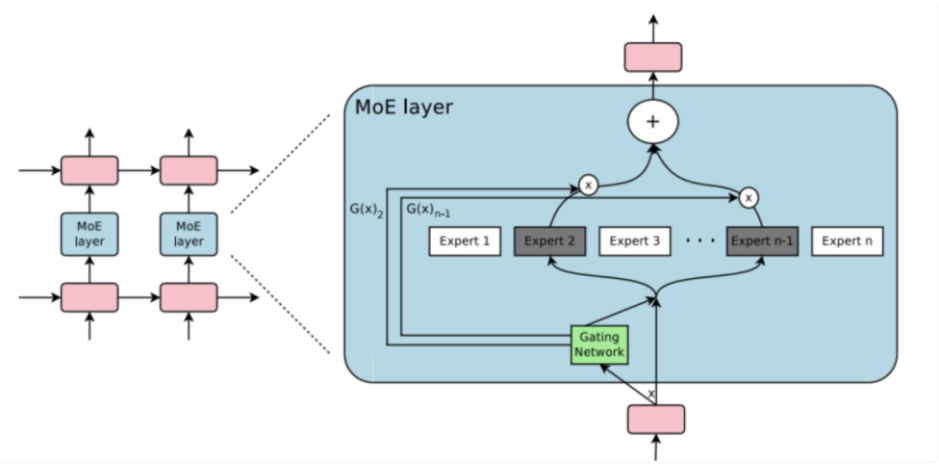
Model Outputs

- **Logits** → Raw unnormalized scores output by model before softmax. One score per vocabulary token.
- **Hidden States** → Intermediate representations at each transformer layer.

- **Activations** → Outputs after applying nonlinear functions (ReLU, GELU, SwiGLU).

---

10. Mixture of Experts (MoE) Architecture



MoE layer from the Outrageously Large Neural Network paper

Source: https://huggingface.co/blog/moe

Core MoE Concepts

- **Mixture of Experts (MoE)** → Architecture with multiple specialized models (experts) and gating mechanism for dynamic expert selection.
- **Expert** → Individual specialized model/network processing specific data subsets or task aspects independently.
- **Gating Network** → Network determining expert weights/probabilities based on input, controlling expert activation and influence.
- **Soft Gating** → Assigns continuous probability distribution over experts, allowing weighted combinations of multiple experts.
- **Hard Gating** → Deterministic selection of specific experts based on highest scores/probabilities.
- **Sparse Gating** → Activates only small expert subset per input, reducing computational cost while maintaining performance.

MoE Operations & Efficiency

- **Load Balancing** → Ensuring even expert workload distribution to prevent over/under-utilization of specific experts.
- **Dynamic Routing** → Real-time input routing to appropriate experts based on gating network decisions.
- **Capacity Factor** → Hyperparameter controlling number of selected experts or model capacity usage during inference.

- **Conditional Computation** → Broader concept where model parts activate conditionally based on input for computational efficiency.
- **Expert Specialization** → Process where each expert learns to handle particular data types or task aspects effectively.
- **Hierarchical MoE** → Multi-level structure where experts themselves can be MoE models for complex decision-making.

Training & Inference

- **Expert Training** → Phase training individual experts on different data subsets while training gating network simultaneously/subsequently.
- **Computation Graph** → Representation of MoE operations including gating decisions and expert contributions.
- **Scalability** → MoE's ability to handle increasing data/model sizes by efficiently distributing workload across experts.
- **Generalization** → MoE's capacity for good performance on unseen data by leveraging diverse expert specializations.

---

11. Training & Optimization

Training Objectives

- **Language Modeling Loss** → Cross-entropy between predicted and actual next tokens.
- **Gradient Descent** → Iterative optimization adjusting weights to minimize loss.
- **Stochastic Gradient Descent (SGD)** → Uses mini-batches to approximate true gradients efficiently.
- **Adam Optimizer** → Adaptive learning rates with momentum. Standard for transformer training.

Training Techniques

- **Gradient Checkpointing** → Trades computation for memory by recomputing activations during backward pass.
- **Mixed Precision Training** → Uses **FP16** for speed while maintaining **FP32** for numerical stability.
- **Gradient Clipping** → Prevents exploding gradients by limiting gradient norms.

Evaluation Methods

- **Perplexity** → Primary metric for language model quality. Lower = better.

- **BLEU Score** → Measures n-gram overlap for generation tasks.
- **Human Evaluation** → Gold standard but expensive and subjective.

---

12. Advanced Fine-Tuning & Optimization

Fine-Tuning Fundamentals

- **Pre-trained Model** → Model initially trained on large general dataset before task-specific adaptation.
- **Transfer Learning** → Leveraging pre-trained model knowledge for new related tasks, reducing training time and data requirements.
- **Domain Adaptation** → Adjusting models for specific domains (medical, legal, financial) through targeted fine-tuning.
- **Supervised Fine-Tuning** → Using labeled data where correct outputs are provided for input-output mapping.
- **Unsupervised Fine-Tuning** → Using unlabeled data with self-supervised techniques for adaptation.

Training Optimization

- **Learning Rate** → Controls step size during gradient descent optimization. Critical hyperparameter for convergence.
- **Learning Rate Schedule** → Strategy for adjusting learning rate during training (warmup, decay, cosine annealing).
- **Batch Size** → Number of training examples processed simultaneously, affecting stability and convergence speed.
- **Epoch** → Complete pass through entire training dataset during fine-tuning process.
- **Adam Optimizer** → Popular adaptive learning rate algorithm combining advantages of AdaGrad and RMSProp.
- **Gradient Accumulation** → Accumulating gradients over multiple batches for effective larger batch sizes.

Regularization & Generalization

- **Early Stopping** → Halting training when validation performance stops improving to prevent overfitting.
- **Overfitting** → Model performs well on training data but poorly on unseen data due to memorization.

- **Underfitting** → Model too simple to capture underlying patterns, performing poorly on both training and validation.
- **Validation Set** → Data subset for monitoring performance and tuning hyperparameters during training.
- **Test Set** → Separate data for final model evaluation on completely unseen examples.
- **Cross-Validation** → Technique dividing data into multiple subsets for robust performance evaluation.

Advanced Techniques

- **Layer Freezing** → Keeping certain pre-trained layers unchanged while fine-tuning others to retain learned features.
- **Feature Extraction** → Using pre-trained model to extract features as input for downstream tasks.
- **Catastrophic Forgetting** → Loss of pre-trained knowledge during fine-tuning, mitigated by gradual unfreezing.
- **Model Distillation** → Training smaller "student" model to replicate larger "teacher" model behaviour for compression.
- **Data Augmentation** → Creating modified versions of training data to increase dataset size and improve generalization.
- **Multi-Task Learning** → Training on multiple tasks simultaneously to leverage shared representations.
- **Ensemble Methods** → Combining predictions from multiple models to improve overall performance and robustness.

Specialized Fine-Tuning Methods

- **Matrix Factorization** → Mathematical technique decomposing matrices into smaller components, used in LoRA.
- **Rank (Linear Algebra)** → Number of linearly independent rows/columns in matrix; lower rank means fewer parameters.
- **Adapter Layers** → Small additional layers inserted into pre-trained models for efficient task adaptation.
- **Mixed-Precision Training** → Using different numerical precisions (FP32, BF16) for different model parts to balance efficiency and accuracy.

Numerical Precision Formats

- **bfloat16 (Brain Float)** → Google's 16-bit format maintaining FP32 exponent size while reducing mantissa precision.

- **nf4 (Normalized Float 4)** → 4-bit floating-point format with normalized values for extreme compression scenarios.
- **Half-Precision (float16)** → Standard 16-bit floating-point with lower precision than float32.
- **Fixed-Point Arithmetic** → Alternative to floating-point using fixed decimal positions, used in some quantization schemes.
- **Quantization Aware Training (QAT)** → Simulating quantization during training to prepare model for post-training quantization.
- **Post-Training Quantization (PTQ)** → Applying quantization to trained model without retraining for deployment efficiency.

---

13. Data & Training Pipeline

Data Processing

- **Data Collection** → Gathering text from web crawls, books, papers, conversations.
- **Data Cleaning** → Removing duplicates, filtering quality, handling PII.
- **Data Deduplication** → Removing exact and near-duplicate content.
- **Data Filtering** → Quality scoring, language detection, content filtering.
- **Tokenization Pipeline** → Converting raw text to model-ready token sequences.

Training Infrastructure

- **Distributed Training** → Coordinating training across multiple machines/GPUs.
- **Checkpointing** → Saving training state for recovery and evaluation.
- **Learning Rate Scheduling** → Adjusting learning rates during training.
- **Warmup** → Gradually increasing learning rate at training start.
- **Early Stopping** → Halting training when validation performance plateaus.

Data Formats

- **JSONL** → JSON Lines format for storing training examples.
- **Parquet** → Columnar storage format for efficient data processing.
- **TFRecord** → TensorFlow's binary format for training data.
- **Arrow** → In-memory columnar format for fast data processing.

L3- Applications & Evaluation

14. GenAI Applications & Techniques

Prompting Strategies

- **Prompt** → Input text guiding model behaviour (instructions, context, examples).
- **Prompt Engineering** → Crafting effective prompts for desired outputs.
- **Zero-Shot** → Performing tasks without task-specific examples in prompt.
- **Few-Shot** → Including 1-5 examples of desired input-output pairs.
- **Chain-of-Thought** → Prompting models to show reasoning steps.
- **Self-Consistency** → Sampling multiple CoT outputs and choosing the most consistent answer.
- **ReAct (Reason + Act)** → Interleaving reasoning steps with external tool/API calls.
- **Tree-of-Thought (ToT)** → Exploring multiple reasoning paths like a search tree.
- **Meta-Prompting** → Prompts that guide how other prompts should be structured (used in multi-agent or tool-using setups).
- **Instruction Tuning / System Prompts** → Using high-level instructions to align model behaviour globally (e.g., "You are a helpful assistant").

Advanced Methods

- **RAG (Retrieval-Augmented Generation)** → Combines external knowledge retrieval with generation to provide factual, up-to-date information.
- **Fine-Tuning** → Training model weights on domain-specific data to adapt for particular tasks or domains.
- **LoRA (Low-Rank Adaptation)** → Parameter-efficient fine-tuning using low-rank matrix decomposition to reduce trainable parameters.
- **QLoRA (Quantized Low-Rank Adaptation)** → Combines quantization with LoRA for even greater memory and compute efficiency.
- **PEFT (Parameter-Efficient Fine-Tuning)** → General category including LoRA, adapters, prefix tuning, layer freezing.
- **Instruction Tuning** → Fine-tuning specifically for following human instructions and commands.

- **RLHF (Reinforcement Learning from Human Feedback)** → Aligns model outputs with human preferences using reward models and human evaluators.

Common Issues

- **Hallucination** → Model generates factually incorrect but plausible-sounding information.
- **Inconsistency** → Model gives contradictory answers to similar or logically related questions (e.g., says "Paris is the capital of France" in one response, but "Lyon is the capital of France" in another).
- **Mode Collapse** → Model produces repetitive or overly similar outputs.
- **Catastrophic Forgetting** → Fine-tuning causes model to forget previously learned knowledge.

---

15. Evaluation & Benchmarking

Capability Evaluation

- **MMLU** → Massive Multitask Language Understanding benchmark.
- **HellaSwag** → Common sense reasoning evaluation.
- **HumanEval** → Code generation benchmark.
- **MATH** → Mathematical problem solving.
- **Big-Bench** → Comprehensive evaluation suite.

Safety Evaluation

- **TruthfulQA** → Measures truthfulness vs. plausible falsehoods.
- **Bias Evaluation** → Testing for demographic and social biases.
- **Toxicity Detection** → Measuring harmful or offensive outputs.
- **Robustness Testing** → Performance under adversarial conditions.

Specialized Metrics

- **ROUGE** → Recall-based evaluation for summarization.
- **BERTScore** → Semantic similarity using BERT embeddings.
- **Exact Match** → Strict string matching for factual questions.
- **F1 Score** → Harmonic mean of precision and recall.

L3.5- Agentic AI Systems

16. Agentic AI Basics

Core Agent Concepts

- **AI Agent** → Autonomous system that perceives environment, makes decisions, and takes actions to achieve specific goals.
- **Agency** → The capacity of an AI system to act independently and make decisions without direct human control.
- **Autonomy** → Degree to which an agent can operate independently without human intervention.
- **Goal-Oriented Behaviour** → Agent's ability to pursue specific objectives through planned sequences of actions.
- **Environment** → External context in which an agent operates (physical world, digital systems, databases, APIs).

Agent Architectures

- **Reactive Agents** → Respond directly to current perceptions without internal state or planning.
- **Deliberative Agents** → Use internal models and planning to make decisions based on goals and reasoning.
- **Hybrid Agents** → Combine reactive and deliberative approaches for both quick responses and strategic planning.
- **Multi-Agent Systems (MAS)** → Multiple agents working together, potentially with different roles and capabilities.
- **Hierarchical Agents** → Structured agent systems with different levels of decision-making and control.

Planning & Reasoning

- **Task Planning** → Breaking down complex goals into executable sequences of actions.
- **Action Selection** → Choosing appropriate actions based on current state and goals.
- **State Space** → All possible configurations or situations an agent can encounter.
- **Search Algorithms** → Methods for finding optimal paths through state spaces (A*, beam search, Monte Carlo tree search).
- **Heuristic Functions** → Rules of thumb guiding search toward promising solutions.

- **Constraint Satisfaction** → Finding solutions that satisfy multiple requirements simultaneously.

Memory & Knowledge Management

- **Working Memory** → Short-term storage for current task-relevant information.
- **Long-Term Memory** → Persistent storage of experiences, knowledge, and learned patterns.
- **Episodic Memory** → Storage of specific experiences and events in temporal sequence.
- **Semantic Memory** → Storage of general knowledge, facts, and concepts.
- **Memory Retrieval** → Process of accessing relevant information from stored memories.
- **Knowledge Graphs** → Structured representations of entities and their relationships for reasoning.

Tool Use & Integration

- **Tool Calling** → Agent's ability to invoke external functions, APIs, or services.
- **Function Calling** → Structured way for LLMs to execute specific functions with parameters.
- **API Integration** → Connecting agents to external services (databases, web services, computation engines).
- **Code Execution** → Agent's capability to write and run code for complex computations or data manipulation.
- **Calculator Integration** → Access to mathematical computation tools for numerical tasks.
- **Web Browsing** → Agent's ability to search, navigate, and extract information from web pages.
- **File System Access** → Reading, writing, and manipulating files and documents.

Communication & Interaction

- **Natural Language Interface** → Human-agent communication using conversational language.
- **Intent Recognition** → Understanding user goals and desired outcomes from natural language.
- **Dialogue Management** → Maintaining coherent conversations across multiple turns.
- **Context Preservation** → Maintaining relevant information across extended interactions.

- **Multi-Modal Interaction** → Processing and responding to text, images, audio, and other input types.

Learning & Adaptation

- **Reinforcement Learning** → Learning through trial and error with reward/penalty feedback.
- **Online Learning** → Adapting behaviour based on new experiences during operation.
- **Meta-Learning** → Learning how to learn new tasks more efficiently.
- **Few-Shot Adaptation** → Quickly adapting to new tasks with minimal examples.
- **Experience Replay** → Reusing past experiences to improve future decision-making.

Advanced Agent Patterns

- **Chain-of-Thought (CoT)** → Step-by-step reasoning process for complex problem solving.
- **Tree of Thoughts** → Exploring multiple reasoning paths simultaneously for better solutions.
- **ReAct (Reasoning + Acting)** → Interleaving reasoning and action-taking for dynamic problem solving.
- **Self-Reflection** → Agent's ability to evaluate its own performance and reasoning.
- **Self-Correction** → Identifying and fixing mistakes in reasoning or actions.
- **Iterative Refinement** → Progressively improving solutions through multiple attempts.

Agent Coordination

- **Task Delegation** → Assigning subtasks to other agents or systems.
- **Collaboration Protocols** → Rules and mechanisms for multi-agent cooperation.
- **Resource Sharing** → Coordinating access to shared computational or information resources.
- **Consensus Mechanisms** → Methods for multiple agents to agree on decisions or facts.
- **Load Balancing** → Distributing work across multiple agents for efficiency.

Evaluation & Metrics

- **Success Rate** → Percentage of tasks completed successfully by the agent.
- **Efficiency Metrics** → Measuring resource usage, time to completion, and cost per task.
- **Safety Metrics** → Evaluating potential harmful actions or unintended consequences.

- **Robustness Testing** → Assessing agent performance under unusual or adversarial conditions.
- **Alignment Assessment** → Measuring how well agent behaviour matches intended goals and values.

Safety & Control

- **Reward Hacking** → Agent exploiting unintended ways to maximize reward signals.
- **Goal Misalignment** → Agent pursuing objectives different from intended human goals.
- **Containment** → Limiting agent's ability to cause harm while allowing beneficial operation.
- **Kill Switch** → Emergency mechanism to immediately halt agent operation.
- **Oversight Mechanisms** → Systems for monitoring and controlling agent behavior.
- **Value Alignment** → Ensuring agent objectives match human values and intentions.

Implementation Frameworks

- **Agent Orchestration** → Coordinating multiple agent components and workflows.
- **Workflow Management** → Managing complex multi-step agent processes.
- **State Management** → Tracking and updating agent internal state across interactions.
- **Error Handling** → Managing failures, exceptions, and recovery in agent systems.
- **Logging & Monitoring** → Tracking agent behaviour, decisions, and performance for analysis.

Specialized Agent Types

- **Research Agents** → Specialized for information gathering, analysis, and synthesis.
- **Code Generation Agents** → Focused on writing, debugging, and optimizing software.
- **Data Analysis Agents** → Specialized for statistical analysis, visualization, and insights.
- **Customer Service Agents** → Designed for user support, troubleshooting, and assistance.
- **Creative Agents** → Focused on content generation, design, and artistic tasks.
- **Scientific Agents** → Specialized for research, hypothesis testing, and scientific reasoning.

Emerging Concepts

- **Swarm Intelligence** → Collective behaviour emerging from groups of simple agents.
- **Emergence** → Complex behaviours arising from interactions of simpler components.

- **Agent-Environment Co-evolution** → Mutual adaptation of agents and their operating environments.
- **Distributed Cognition** → Intelligence distributed across multiple agents and tools.
- **Cognitive Architectures** → Comprehensive frameworks modeling human-like reasoning in agents.

L4- Deployment & Performance

---

17. Runtime & Serving Metrics

Latency Metrics

- **TTFT (Time to First Token)** → Delay between request and first generated token. Includes prompt processing.
- **TPOT (Time per Output Token)** → Average time to generate each subsequent token after the first.
- **Latency** → Total response time from request to completion.

Throughput Metrics

- **Tokens per Second** → Rate of token generation (input + output tokens processed).
- **Requests per Second** → Number of complete requests handled per second.
- **Concurrent Users** → Maximum users served simultaneously.

Efficiency Optimizations

- **KV Cache** → Stores attention keys/values from previous tokens, avoiding recomputation.
- **Batching** → Processing multiple requests simultaneously to improve GPU utilisation.
- **Streaming** → Returning tokens to user as generated rather than waiting for completion.
- **Speculative Decoding** → Uses fast **draft** model to propose tokens, larger model to verify.

---

18. Inference Optimizations

Model Compression

- **Quantization** → Reducing precision (FP32 → INT8/4-bit) for smaller, faster models.

- **Pruning** → Removing less important model weights to reduce model size.
- **Distillation** → Training smaller **student** model to mimic larger **teacher** model.

Memory & Compute Optimization

- **FlashAttention** → Memory-efficient attention implementation reducing memory usage quadratically.
- **Paged Attention** → Virtual memory techniques for very long contexts (used in vLLM).
- **Gradient Checkpointing** → Recompute activations during backward pass to save memory.

Advanced Attention

- **Sparse Attention** → Only compute attention for subset of token pairs.
- **Sliding Window Attention** → Fixed-size attention window for handling long sequences.
- **Multi-Query Attention** → Shares key/value heads across query heads for efficiency.

---

19. Deployment & Infrastructure

Serving Architecture

- **Inference Engine** → Optimised runtime for model serving:
  - **vLLM** → High-throughput LLM serving with PagedAttention.
  - **TensorRT-LLM** → NVIDIA's optimized inference library.
  - **DeepSpeed** → Microsoft's training and inference framework.
  - **ONNX Runtime** → Cross-platform inference optimization.

Scaling Strategies

- **Model Sharding** → Splitting large model weights across multiple GPUs/machines.
- **Tensor Parallelism** → Splitting individual matrix operations across devices.
- **Pipeline Parallelism** → Distributing different layers across different devices.
- **Data Parallelism** → Running identical models on different data batches.

Production Infrastructure

- **Load Balancer** → Distributes incoming requests across multiple inference servers.
- **Autoscaling** → Automatically adjusts number of inference nodes based on demand.
- **Model Registry** → Centralised storage and versioning for model artifacts.

- **A/B Testing** → Comparing different models or configurations in production.
- **Monitoring** → Tracking latency, throughput, cost, and quality metrics.

Memory Management

- **Offloading** → Moving parts of computation to CPU/disk when GPU memory insufficient.
- **Memory Mapping** → Loading model weights on-demand from storage.
- **Checkpointing** → Saving model state for recovery and resumption.

---

20. GPU & Hardware Specifics

GPU Architecture

- **CUDA** → NVIDIA's parallel computing platform enabling GPU programming.
- **Tensor Cores** → Specialised GPU units optimized for matrix multiplications in AI workloads.
- **VRAM (GPU Memory)** → Stores model weights, activations, KV cache. Major bottleneck for large models.
- **Memory Bandwidth** → Speed of reading/writing GPU memory. Critical for throughput.
- **GPU Utilisation** → Percentage of GPU compute actively used during training/inference.

Interconnects

- **PCIe** → Standard interface connecting GPUs to CPU and system memory.
- **NVLink** → High-speed interconnect for GPU-to-GPU communication within nodes.
- **InfiniBand** → High-performance networking for multi-node GPU clusters.

Multi-GPU Training

- **Data Parallelism** → Replicate model across GPUs, split training data.
- **Model Parallelism** → Split large model layers across multiple GPUs.
- **Pipeline Parallelism** → Divide model into sequential stages across devices.
- **ZeRO** → Zero Redundancy Optimizer for memory-efficient distributed training.
- **FSDP** → Fully Sharded Data Parallel for scaling to thousands of GPUs.

Hardware Considerations

- **FP16/BF16 Support** → Hardware acceleration for half-precision training/inference.

- **Memory Hierarchy** → GPU registers → shared memory → global memory → CPU memory → disk.
- **Compute vs Memory Bound** → Whether performance limited by computation or memory access.

L5 - Advanced & Governance

---

21. Advanced Statistical Methods

Bayesian Methods

- **Bayesian Inference** → Incorporating prior beliefs and updating with evidence.
- **Variational Inference** → Approximating complex posteriors with simpler distributions.
- **MCMC (Markov Chain Monte Carlo)** → Sampling from complex probability distributions.
- **Uncertainty Quantification** → Measuring model confidence in predictions.

Advanced Evaluation

- **Hypothesis Testing** → Statistical significance testing for model comparisons.
- **Confidence Intervals** → Range estimates for model performance metrics.
- **Bootstrap Sampling** → Resampling technique for robust evaluation estimates.
- **Cross-Validation** → Splitting data to assess generalization performance.

Information Theory Applications

- **Mutual Information** → Measuring shared information between variables.
- **Information Gain** → Reduction in uncertainty from observing additional variables.
- **Rate-Distortion Theory** → Theoretical limits of lossy compression.

---

22. Security & Safety Concepts

AI Safety

- **Alignment** → Ensuring AI systems pursue intended goals and values.
- **Constitutional AI** → Training models using a set of principles/constitution.
- **Red Teaming** → Systematic attempts to find harmful model behaviors.

- **Jailbreaking** → Techniques to bypass safety guardrails and restrictions.
- **Prompt Injection** → Malicious prompts designed to manipulate model behavior.

Security Concerns

- **Model Extraction** → Attempting to steal model weights or architecture through API access.
- **Membership Inference** → Determining if specific data was used in training.
- **Data Poisoning** → Introducing malicious data to corrupt model behavior.
- **Adversarial Examples** → Inputs designed to fool model predictions.
- **Backdoor Attacks** → Hidden triggers that cause specific malicious behaviors.

---

23. Cost & Economics

Training Costs

- **Compute Cost** → GPU/TPU hours for training (millions of dollars for large models).
- **Energy Consumption** → Power usage and carbon footprint.
- **Data Acquisition** → Licensing, web crawling, human annotation costs.
- **Human Feedback** → RLHF annotation and preference collection.

Inference Costs

- **Per-Token Pricing** → Cost models based on input/output token counts.
- **Fixed vs Variable Costs** → Infrastructure vs. usage-based pricing.
- **Cost Per Query** → End-to-end cost including serving infrastructure.
- **Cost Optimization** → Techniques to reduce serving costs.

---

24. Regulatory & Compliance

AI Governance

- **AI Act (EU)** → Regulatory framework for AI systems.
- **Model Cards** → Documentation of model capabilities, limitations, biases.
- **Data Governance** → Policies for data collection, usage, retention.
- **Algorithmic Auditing** → Systematic evaluation of AI system fairness.

Privacy & Rights

- **GDPR Compliance** → European data protection requirements.
- **Right to Explanation** → Users' rights to understand AI decisions.
- **Data Subject Rights** → Deletion, portability, correction of personal data.
- **Consent Management** → Handling user permissions for data usage.

---

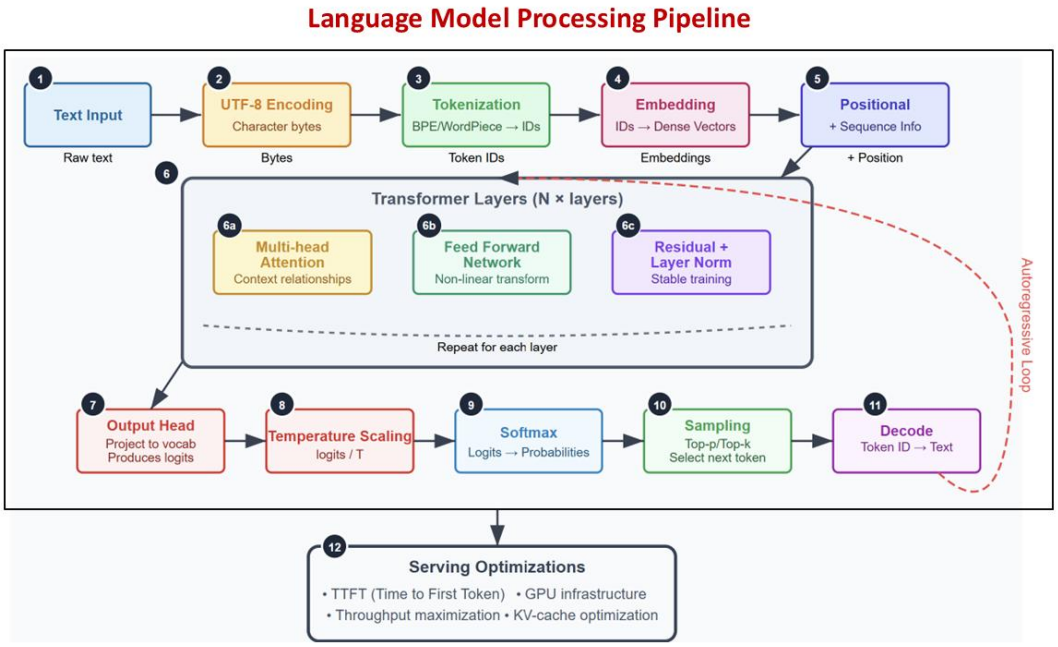
25. Research Frontiers

Research Areas

- **Mechanistic Interpretability** → Understanding internal model computations.
- **Scaling Laws** → Mathematical relationships between compute, data, performance.
- **Emergent Abilities** → Capabilities appearing at certain model scales.
- **Transfer Learning** → Applying knowledge from one domain to another.
- **Meta-Learning** → Learning to learn new tasks quickly.

L6- End-to-End Flow

26. Language Model Processing Flow: Text → Tokens → Model → Output



1. Text Input → UTF-8 Encoded Characters

The process begins when raw text is received and converted into UTF-8 encoding, which represents each character as one or more bytes. This standardized encoding ensures consistent handling of international characters, emojis, and special symbols across different systems.

2. Tokenization → BPE/WordPiece Converts to Token IDs

The UTF-8 text undergoes tokenization using algorithms like **Byte Pair Encoding (BPE) or WordPiece**. These methods break text into subword units that balance vocabulary size with meaningful representation. Common words might be single tokens, while rare words are split into smaller components. Each token is assigned a unique numerical ID from the model's vocabulary (typically 32K-100K tokens).

3. Embedding → Token IDs Become Dense Vectors

Token IDs are mapped to high-dimensional dense vectors (typically 768-4096 dimensions) through an embedding matrix. These learned representations capture semantic relationships between tokens, where similar tokens have similar vector representations in the embedding space.

4. Positional Encoding → Adds Sequence Order Information

Since transformers process all tokens simultaneously, positional encodings are added to embeddings to preserve sequence order information. These can be sinusoidal functions or learned embeddings that help the model understand token positions and relationships within the sequence.

5. Transformer Layers → Multi-head Attention + FFN Process Representations

The core processing occurs through multiple transformer layers (12-96+ layers in large models). Each layer contains:

- **Multi-head Self-Attention**: Allows tokens to attend to other tokens in the sequence, capturing long-range dependencies and contextual relationships
- **Feed-Forward Network (FFN)**: Applies non-linear transformations to process the attended representations
- **Residual connections and layer normalization** ensure stable training and information flow

6. Output Head → Projects to Vocabulary Size, Produces Logits

The final transformer layer's output is projected through a linear layer that maps the hidden dimension back to vocabulary size, producing raw scores (logits) for each possible next token.

7. Temperature Scaling → Controls Randomness of Logits

BEFORE softmax, the raw logits are divided by the temperature parameter (T). This critical step controls the randomness of the output:

- **Lower temperature (T < 1.0)**: Makes logits more extreme → sharper probability distribution → more deterministic/focused outputs
- **Higher temperature (T > 1.0)**: Makes logits less extreme → flatter probability distribution → more random/creative outputs
- **Temperature = 1.0**: No change to the original logits

8. Softmax → Converts Temperature-Scaled Logits to Probabilities

The temperature-scaled logits are converted to probability distributions using the softmax function, ensuring all probabilities sum to 1. This creates a distribution over the entire vocabulary for the next token prediction.

9. Sampling → Select Next Token Using Top-p/Top-k Strategies

The next token is selected from the probability distribution using various sampling strategies:

- **Top-k**: Only considers the k most likely tokens
- **Top-p (nucleus)**: Considers tokens until cumulative probability reaches p

- **Greedy**: Always selects the highest probability token

10. Decode → Convert Token IDs Back to Text

The selected token ID is converted back to its corresponding text representation using the tokenizer's vocabulary mapping, then concatenated with the existing output sequence.

11. Serving Optimizations → Distributed Across Pipeline

Production optimizations are applied throughout the pipeline:

- **Tokenization**: Batch processing multiple requests
- **Attention**: KV-cache storage, layer fusion
- **Generation**: Speculative decoding, parallel sampling
- **Infrastructure**: GPU scheduling, resource management
- **Memory/TTFT**: Efficient allocation, fast first token delivery