How to Be Better Than Most in GenAI

Contents

Core LLM Building Blocks

1. **Transformer Architecture** → Core design (encoder-only like BERT, decoder-only like GPT/Claude/Gemini etc [autoregressive], encoder-decoder like T5)

2. **Tokenization** → Break text into tokens and map them to numbers

3. **Embedding Spaces** → Map tokens into a semantic space where closer vectors mean similar meaning

4. **Positional Encoding** → Adds sequence order information to token embeddings

5. **Attention** → Highlights the most relevant tokens in context

6. **Self-Attentio**n → Each token attends to every other token for context

7. **Cross-Attention** → Connect encoder and decoder (in encoder-decoder models)

8. **Multi-Head Attention** → Several attention heads capture different patterns in parallel

9. **Feed-Forward Networks** → Nonlinear layers that transform representations between attention blocks

10. **Residual Connection**s → Shortcut links that preserve signals and help gradient flow

11. **Layer Normalization** → Normalizes activations to stabilize and speed up training

12. **Output Projection (LM Head)** → Final linear layer mapping hidden states into logits

13. **Logits** → Raw prediction scores for each token before probabilities

14. **Softmax** → Turns logits into a probability distribution

15. **Sampling from Probabilities** → Chooses the next token based on probability weights

16. **RoPE** → Rotary Positional Encoding, adds relative position info for longer sequences

17. **ALiBi / Relative Positional Encoding** → Alternative to RoPE for long contexts

18. **Linear / Performer Attention** → Efficient attention variants for very long sequences

19. **Grouped Query Attention (GQA)** → Reduces memory use by sharing keys/values across heads

20. **Multi-Head Latent Attention** → Extends attention into hidden spaces for richer context

21. **SwiGLU/GeLU Activations** → Smooth nonlinear functions that improve expressiveness

22. **RMSNorm** → Root-mean-square normalization, a lighter alternative to LayerNorm

---

Training & Tuning

1. **Pretraining** → Build general world knowledge from large datasets

2. **Mixed Precision Training** → Speed up training with lower-precision arithmetic

3. **Sharded / Distributed Training** → Scale across multiple GPUs/nodes

4. **Continual / Lifelong Learning** → Update models without forgetting old knowledge

5. **Self-Instruction / Self-Play** → AI generates its own training examples

6. **Fine-Tuning** → Adapt the model for specific domains or tasks

7. **Supervised Fine-Tuning (SFT)** → Train on curated input-output pairs

8. **LoRA** → Parameter-efficient adapters for cheap fine-tuning

9. **QLoRA** → LoRA + quantization, enabling fine-tuning of huge models on modest hardware

10. **PEFT** → Family of methods (e.g., LoRA, QLoRA, adapters) updating only small parts of the model

11. **Instruction Tuning** → Teach models to follow natural language instructions

12. **RLHF** → Align model outputs with human preferences via feedback

13. **Constitutional AI** → AI-written principles for safer alignment

14. **DPO / PPO / GRPO** → Optimization algorithms for preference alignment

15. **Distillation** → Transfer knowledge from a large model into a smaller one

16. **Gradient Descent & Backpropagation** → Core optimization mechanics

17. **Loss Functions** → Cross-entropy loss, perplexity, etc. that guide learning

18. **Learning Rate Scheduling** → Adjust training speed for stability

19. **Batch Size & Gradient Accumulation** → Control efficiency and memory use

20. **Scaling Laws** → Predictable links between size, data, and performance

21. **Mixture-of-Experts Fine-Tuning** → Activate only subsets of parameters for efficiency

22. **Continual / Lifelong Learning** → Update models without forgetting past knowledge

23. **Preference Optimization (IPO, KTO, etc.)** → Newer RLHF alternatives without full RL

Generation Controls

1. **Entropy** → Measures uncertainty, higher entropy = more diverse but less predictable outputs

2. **Temperature** → Controls randomness; higher = creative, lower = precise

3. **Top-k / Top-p** → Sampling filters, Highter = safer, looser = more diverse

4. **Repetition Penalty** → Discourages loops; too strong may block valid words

5. **Beam Search** → Finds higher-probability outputs, safer but less creative

6. **Stop Sequences** → Force model to end at defined markers, too strict may cut off early

7. **Context Window** → Defines memory size, longer = more context, but costly

8. **Chain-of-Thought** → Stepwise reasoning, improves logic but can slow generation

9. **Speculative Decoding** → Drafts with a smaller model, faster but may need corrections

10. **Contrastive Decoding** → Balances fluency vs factuality

11. **Medusa / Speculative Multi-Path Decoding** → Multi-hypothesis generation for speed + accuracy

---

Knowledge & Retrieval

1. **RAG** → Combine LLMs with external knowledge sources for up-to-date answers

2. **Vector Databases** → Store embeddings and perform fast similarity search

3. **In-Context Learning** → Model adapts using examples provided in the prompt

4. **Knowledge Graphs** → Represent facts as structured entities and relationships

5. **Semantic Search** → Retrieve information based on meaning, not exact keywords

6. **Few-Shot Learning** → Learn from a small number of examples

7. **Zero-Shot Learning** → Generalize to new tasks without any examples

8. **Hybrid Search** → Combine semantic, keyword, and graph-based retrieval

9. **Retrieval Augmentation via Loops** → Iteratively refine queries to improve results

10. **Reranker** → Re-rank retrieved results to prioritize the most relevant content

11. **Dense Passage Retrieval (DPR)** → Embedding-based retrieval for large corpora

12**. Sparse Retrieval / BM25** → Keyword-based retrieval for efficiency and baseline

13. **Feedback-Augmented Retrieval** → Use human or model feedback to improve relevance

14. **Chain-of-Retrieval** → Multi-step retrieval pipelines for complex queries

15. **Retriever-Generator Loops** → Iterative retrieval + generation for difficult queries

16. **Hybrid Dense-Sparse Retrieval** → Combine semantic + keyword retrieval for better accuracy

17. **Multi-hop Reasoning** → Retrieve multiple connected facts to answer complex queries

18. **Embedding Updates / Incremental Indexing** → Keep retrieval database fresh efficiently

---

Efficiency & Scaling

1. **Quantization** → Compress model size and memory usage with minimal accuracy loss
2. **Sparse Models / MoE** → Activate only relevant experts to save compute
3. **Model Parallelism** → Distribute large models across multiple devices
4. **Data Parallelism** → Process multiple batches simultaneously for faster training
5. **Gradient Checkpointing** → Save memory by recomputing intermediate activations on demand
6. **KV Caching** → Cache key-value pairs to accelerate inference
7. **Latency vs. Cost Tradeoff** → Larger models = slower & costlier, smaller = faster & lighter
8. **Model Serving** → Efficient deployment using batching and request queuing
9. **Edge Deployment** → Run models on local or mobile devices with limited resources
10. **Mixed Precision Training** → Use FP16/BF16 to speed up training and reduce memory
11. **Sharded / Distributed Training** → Split model parameters across devices for massive models
12. **Pipeline Parallelism** → Overlap sequential layer computation across devices for speed
13. **Activation & Parameter Offloading** → Move parts of model to CPU to save GPU memory
14. **Sparse Attention** → Compute attention only for relevant tokens, reducing cost
15. **Memory-Mapped Checkpoints** → Load large models efficiently without full RAM usage
16. **Elastic / Dynamic Batching** → Adjust batch size dynamically to optimize throughput
17. **Inference Optimization** → Operator fusion, kernel tuning, and caching for faster runtime
18. **Quantization-Aware Training (QAT)** → Fine-tune models with quantization to retain accuracy

---

Data & Preprocessing

1. **Data Cleaning & Filtering** → Ensure high-quality, relevant, and consistent training data
2. **Tokenizer Training** → Build vocabularies using BPE, SentencePiece, or Unigram models
3. **Data Deduplication** → Remove repeated or near-duplicate examples to improve learning
4. **Data Mixing & Curriculum Learning** → Present data strategically for better convergence
5. **Data Augmentation** → Expand dataset with synthetic or modified examples
6. **Synthetic Data Generation** → AI-generated data to fill gaps or rare cases

7. **Balanced Sampling** → Ensure diverse representation across domains/classes

---

Evaluation & Benchmarks

1. **Perplexity** → Measures how well a model predicts text (core LM metric)

2. **BLEU / ROUGE / BERTScore** → Compare generated text to reference quality

3. **Benchmark Suites** → Standardized tests like MMLU, HellaSwag, BIG-bench for model evaluation

4. **Human Evaluation** → Collect human judgments for accuracy, coherence, and safety

5. **Factuality / Truthfulness Metrics** → Specialized evaluation for hallucination-prone outputs

6. **Consistency / Contradiction Metrics** → Check if model outputs are logically consistent across queries

7. **Bias & Fairness Metrics** → Quantify demographic or cultural biases in outputs

8. **Adversarial Robustness** → Test resilience to malicious or tricky prompts

9. **Knowledge Probing** → Evaluate stored factual knowledge (e.g., LAMA, TruthfulQA)

10. **Efficiency & Cost Metrics** → Measure latency, throughput, memory, and compute requirements

11. **Explainability / Interpretability Evaluation** → Assess clarity and transparency of model reasoning

---

Extensions

1. **Multimodality** → Combine text, images, audio, and video for richer understanding

2. **Agents** → LLMs that plan, reason, and take actions autonomously

3. **Agentic AI** → LLMs with autonomous decision-making, memory, and goal-oriented behaviour

4. **Multi-Agent Systems** → Teams of LLMs with specialized roles for complex tasks

5. **Tool Use / Function Calling** → LLMs interact with APIs, databases, and external tools

6. **Prompt Engineering** → Design inputs to guide models toward better outputs

7. **Few-Shot & One-Shot Learning** → Adapt from minimal examples for quick generalization

8. **Auto-Prompting / Self-Instruction** → Models generate their own prompts to improve learning

9. **Chain of Thought (CoT)** → Break problems into step-by-step reasoning

10. **ReAct** → Combine reasoning with tool use and actions

11. **Self-Consistency** → Compare multiple reasoning paths → pick best answer

12. **Tree of Thoughts (ToT)** → Explore many reasoning branches before deciding

13. **Reflexion** → Model critiques, learns, and refines its own answers

14. **Memory** → Agents recall past interactions for long-term context

---

Safety & Limits

1. **Hallucination** → Fluent but incorrect or nonsensical outputs

2. **Alignment** → Ensure model behavior is safe, ethical, and policy-compliant

3. **Guardrails** → Rule-based or learned filters to prevent harmful outputs

4. **Bias & Fairness** → Detect and mitigate demographic or cultural prejudices

5. **Privacy & Data Leakage** → Protect sensitive information during training and inference

6. **Adversarial Attacks** → Malicious inputs designed to mislead or exploit the model

7. **Interpretability** → Understand how and why the model makes decisions

8. **Calibration / Uncertainty Estimation** → Avoid overconfident wrong predictions

9. **Red-Teaming** → Stress-test models for safety, robustness, and alignment

10 **Robustness Evaluation** → Test model resilience against noise, domain shifts, and edge cases