Attention Heads, MLPs, and Circuits: A Naming Convention for Transformer Mechanisms

Karol Kowalczyk

November 2025

Abstract

Transformer models implement computation through three levels of organization: attention heads route information, MLP layers store and transform features, and multi-component circuits integrate these primitives into complex behaviors. Mechanistic interpretability research has identified numerous mechanisms—induction, factual recall, coreference resolution, safety enforcement—but naming remains inconsistent. The same mechanism appears under different names; the same name refers to different mechanisms; descriptions conflate what a mechanism computes with which components implement it.

I propose a mechanism-first naming convention: describe the computational transformation independently, then specify which components (attention heads, MLPs, circuits, sparse autoencoder features, or architectural elements) implement it. This taxonomy organizes mechanisms into nine functional stacks: Pattern Completion, Memory Recall, Information Routing, Feature Transformation, Composition, Safety Control, Output Generation, Reasoning, and Infrastructure. Each mechanism entry specifies primary implementation, behavioral signatures, ablation effects, and component-level details. This framework unifies attention head, MLP, and circuit perspectives under consistent terminology.

Contents

1 Introduction

1.1 Motivation

Transformers achieve remarkable performance through layered computation: attention heads route information selectively, MLP layers store knowledge and transform features, and circuits combine these primitives across depths. Mechanistic interpretability research seeks to reverse-engineer these computations into human-understandable mechanisms [5, 11, 13].

Research has identified specialized mechanisms: induction heads implement pattern completion [11], MLP layers function as key-value memories [7], the IOI circuit performs indirect object identification through 26 attention heads [13], and sparse autoencoders extract monosemantic features from superposed representations [2]. Yet naming remains fragmented. The same mechanism appears as "induction head," "pattern head," "copy head," and "ICL head." Different mechanisms share names: "copy head" refers to both induction and name-mover behaviors. Descriptions conflate mechanism (what is computed) with component (how it is implemented).

1.2 The Problem: Component-First Thinking

Current naming conventions organize by component type: attention head taxonomies [14], MLP neuron studies [7, 4], circuit analyses [13]. This component-first approach creates three problems. First, the same computational function appears under different names when implemented differently. Second, component-specific vocabularies prevent cross-component integration. Third, circuit-level understanding remains disconnected from component-level descriptions.

1.3 The Solution: Mechanism-First Naming

I propose mechanism-first organization: describe what is computed, then specify which components implement it. Each mechanism entry includes: (1) computational transformation (component-agnostic), (2) primary implementation (attention heads, MLPs, circuits, SAE features, or architecture), (3) component-specific details (attention patterns, MLP structure, circuit composition), (4) behavioral signatures (observable effects), and (5) ablation patterns (what breaks when removed).

This framework accommodates multiple implementation modes. Pattern completion combines attention heads (previous-token, induction) with MLP neurons (n-gram storage). Factual recall integrates entity heads (query identification), MLP neurons (knowledge storage), and name-mover heads (output routing). Safety enforcement requires multi-stage circuits (detection, policy, refusal) spanning early to final layers.

1.4 Scope and Limitations

This taxonomy focuses on well-documented mechanisms with empirical support. It describes current understanding while remaining flexible for future discoveries. The taxonomy acknowledges three key limitations: (1) mechanisms are polyfunctional—components contribute to multiple

computations, (2) implementation varies across architectures—GPT, LLaMA, and Claude show mechanistic similarities but architectural differences, and (3) understanding is incomplete—many MLP functions and circuit interactions remain poorly understood.

1.5 Contributions

This work provides: (1) mechanism-first taxonomy organizing transformer computation into nine functional stacks, (2) standardized naming bridging attention head, MLP, and circuit perspectives, (3) implementation specifications detailing how different components realize each mechanism, (4) cross-reference tables mapping literature terms to canonical mechanism names, and (5) empirically grounded descriptions synthesizing interpretability research.

1.6 Document Structure

Section 2 reviews mechanistic interpretability and component types. Section 3 explains mechanism identification methods. Section 4 introduces the depth model. Sections 5.1 through 5.9 catalog mechanisms by functional stack. Section 6 analyzes cross-stack patterns and limitations. Section 7 concludes with future directions.

2 Background

2.1 Transformer Architecture

Transformers [12] process sequences through alternating attention and MLP layers. Each layer reads from and writes to a residual stream via additive contributions. Attention heads route information using learned query-key-value patterns. MLP layers transform features through expansion ($d \to 4d$), nonlinear activation, and contraction ($4d \to d$). Layer normalization stabilizes activations. Residual connections enable gradient flow through depth.

2.2 Three Levels of Organization

Transformer computation operates at three levels. **Attention heads** implement specialized routing patterns: previous-token heads create shifted representations, induction heads match patterns, name-mover heads copy entities to output positions [5, 11]. **MLP layers** store knowledge as key-value memories [7, 8]: first-layer weights detect patterns (keys), second-layer weights provide associations (values). Knowledge neurons encode specific facts [4, 9]. **Circuits** combine attention and MLP across layers: the IOI circuit uses duplicate-token heads, S-inhibition heads, name-mover heads, and MLPs to resolve indirect objects [13].

2.3 Sparse Autoencoders and Superposition

Neurons exhibit polysemanticity: single neurons respond to multiple unrelated concepts [6]. Superposition enables models to represent more features than available dimensions by encoding

features as sparse linear combinations [6]. Sparse autoencoders (SAEs) extract monosemantic features by learning overcomplete sparse bases (10–100× model dimensions) [2, 3]. SAE features enable fine-grained interpretability but do not automatically explain feature interactions.

2.4 Why Naming Matters

Inconsistent naming hinders communication, replication, and integration. Researchers describe the same mechanism differently: "induction head" vs. "pattern head" vs. "ICL head." Different mechanisms share names: "copy head" refers to induction, name-mover, and duplicate-token behaviors. Circuit studies reference components by position rather than function. This taxonomy provides stable vocabulary spanning attention heads, MLPs, and circuits.

3 Methodology

3.1 Mechanism Identification

Mechanisms are identified through converging evidence: (1) **Attention pattern analysis** reveals specialized routing (diagonal for previous-token, content-based for induction) [5], (2) **Ablation studies** show behavioral changes when components removed [13], (3) **Activation patching** traces information flow through specific paths [13], (4) **Logit attribution** identifies components contributing to predictions [10], and (5) **SAE feature analysis** extracts interpretable concepts [2].

3.2 Mechanism vs. Component

This taxonomy distinguishes mechanism (computational transformation) from component (implementation substrate). The induction mechanism detects [A][B]...[A] patterns and predicts [B]. This mechanism is implemented by: previous-token heads (create shifted representations), induction heads (match patterns via attention), and MLP neurons (store common bigrams). Separating mechanism from implementation enables cross-architecture comparison and multi-component integration.

3.3 Inclusion Criteria

Mechanisms are included when: (1) independently observed across multiple models, (2) behaviorally validated through ablations, (3) mechanistically understood at component level, and (4) reproducible using standard interpretability techniques. Proposed mechanisms lacking empirical support are marked clearly.

4 Depth Model

4.1 Four-Level Depth Organization

Mechanisms concentrate at predictable depths [5, 13]: **Early (E, 0.00–0.25)** layers process surface features—delimiters, local patterns, content detection. **Middle (M, 0.25–0.70)** layers implement core computation—induction, factual recall, reasoning. **Late (L, 0.70–0.88)** layers perform integration—entity movement, topic routing, strategy planning. **Final (F, 0.88–1.00)** layers enforce constraints—safety, formatting, completion control.

4.2 Relative Depth

Depth is expressed as fraction of total layers for architecture-independent comparison. A mechanism at relative depth 0.40 occupies similar functional space in 12-layer and 96-layer models. This enables cross-model generalization while acknowledging depth-specific variations.

4.3 Depth Boundaries

Depth categories have defined boundaries but mechanisms may span ranges. MLP knowledge storage operates at all depths with hierarchical specialization: early layers store syntax, middle layers store facts, late layers store semantic concepts [8]. Sparse autoencoder features span depths with varying abstraction levels [2].

5 Mechanism Catalog

This section catalogs mechanisms by functional stack. Each entry specifies: mechanism description (what is computed), primary implementation (which components), implementation details (how components realize it), ablation effects (what breaks), and examples (input→mechanism→output).

5.1 Pattern Completion Stack

Stack overview: Detect and complete patterns from context. Enable in-context learning through pattern matching and repetition detection. Support sequence continuation and algorithmic behavior.

5.1.1 Induction

Depth: `0.25-0.55` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *induction mechanism, pattern completion, in-context learning*

Detect and complete patterns of form [A][B]...[A] → predict [B]. Core mechanism for in-context learning and few-shot adaptation. Requires multi-component implementation: previous-token heads create shifted representations, induction heads match patterns via attention, MLP neurons store common bigrams. Enables pattern-based prediction without parameter updates.

Attention Heads: Previous-token heads (E, 0.05–0.20) attend to position $i-1$ with uniform offset pattern. Induction heads (M, 0.25–0.55) attend to previous occurrences of current token content.
MLP Layers: N-gram storage neurons (E-M, 0.15–0.55) store frequent bigram and trigram patterns as key-value associations.
Circuit: Previous-token → Induction → MLP retrieval working in composition across 3–8 layers.

Expected ablation: Severe loss of in-context learning capability. Major degradation on few-shot tasks. Loss of pattern completion and sequence continuation. Reduced ability to learn from examples in prompt.

Example

Input: "When Mary and John went to the store, Mary gave milk to John. When Susan and Bob went to the store, Susan gave milk to..."
Behavior: Detect pattern: [A] gave milk to [B] → predict [B]
Effect: Output "Bob" by analogical pattern matching

Status: Well-documented | **Related:** duplicate-token, bigram-continuation, algorithmic-continuation

5.1.2 Duplicate Token Detection

Depth: `0.30-0.58` | **Primary Implementation:** Attention heads | **Literature names:** *duplicate detection, repetition detection, token matching*

Detect and track repeated tokens or entities across arbitrary distances. Distinguish first occurrence from subsequent mentions. Support induction mechanism by identifying repeated elements. Enable token-level pattern recognition independent of content. Component of multiple circuits including IOI (indirect object identification) and entity tracking.

Attention Heads: Duplicate-token heads (M, 0.30–0.58) perform exact token identity matching, attending from repeated token to its first occurrence.
Circuit: Feeds into induction heads, S-inhibition heads, and name-mover heads as repetition signal.

Expected ablation: Moderate degradation in pattern matching and entity tracking. Notable loss on tasks requiring repetition detection. Reduced support for induction mechanism and entity circuits.

Example

Input: "The cat sat on the mat. The cat..."
Behavior: Second "cat" detects first "cat" as duplicate
Effect: Signal repetition for downstream processing

Status: Well-documented | **Related:** induction, entity-tracking, S-inhibition

5.1.3 Bigram Continuation

Depth: `0.15-0.52` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *n-gram prediction, statistical continuation, phrase completion*

Continue common bigram and n-gram patterns using statistical knowledge from training. Predict next token based on immediate local context (1–3 tokens). Implement fast, parameter-based pattern matching for frequent sequences. Enable fluent continuation of idioms, common phrases, and formulaic expressions. Complement induction with statistical frequency information.

MLP Layers: N-gram storage neurons (E-M, 0.15–0.52) encode frequent sequences. First sublayer: detect n-1 token context (key). Second sublayer: provide continuation options (value).
Attention Heads: N-gram heads (M, 0.28–0.52) attend to matching contexts to retrieve stored patterns.

Expected ablation: Moderate loss of fluent continuation for common phrases. Noticeable degradation on completion of idioms and formulaic expressions. Reduced statistical coherence in generation. Increased reliance on attention-based pattern matching.

Example

Input: "Once upon a..."
Behavior: Match "once upon a" bigram pattern, retrieve high-frequency continuation
Effect: Output "time" as statistically likely completion

Status: Observed | **Related:** induction, algorithmic-continuation

5.1.4 Algorithmic Continuation

Depth: `0.35-0.65` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *sequence continuation, rule following, mathematical pattern completion*

Continue algorithmic sequences by detecting underlying rules: counting, arithmetic progressions, alphabetic sequences, mathematical patterns. Extract systematic rules from examples and apply consistently. Support mathematical and logical pattern completion beyond simple memorization. Enable rule-based generation without explicit instruction. Integration point between pattern matching and reasoning mechanisms.

Attention Heads: Algorithmic heads (M, 0.35–0.60) detect regular patterns and systematic relationships in sequences.
MLP Layers: Rule storage neurons (M, 0.40–0.65) encode mathematical operations and sequence transformation rules.
Circuit: Integrates with reasoning mechanisms (M-L) for complex rule application.

Expected ablation: Significant loss of algorithmic continuation ability. Major degradation on sequence completion tasks. Reduced performance on mathematical and logical patterns. Loss of systematic rule application.

Example

Input: "2, 4, 8, 16, 32, ..."

Behavior: Detect doubling pattern, apply rule systematically

Effect: Output "64" (next power of 2)

Status: Observed | **Related:** induction, reasoning-integration

5.1.5 Position-Based Copying

Depth: `0.05-0.25` | **Primary Implementation:** Attention heads | **Literature names:** *previous-token attention, offset attention, adjacent copying*

Create positionally-shifted representations by attending to fixed offsets (typically position $i-1$). Foundational primitive enabling bigram processing and pattern detection. Implement constant offset attention independent of content. Enable MLPs to process adjacent token pairs as input. Essential building block for induction and other pattern-based mechanisms.

Attention Heads: Previous-token heads (E, 0.05–0.20) implement nearly uniform attention to position $i-1$ from position i. Content-independent, purely positional pattern.
Circuit: First stage of induction circuit and many pattern-matching circuits.

Expected ablation: Significant degradation in pattern completion and local context processing. Loss of bigram processing capability. Severe accuracy drop on induction tasks. Reduced ability to continue sequences requiring adjacent token information.

Example

Input: "...A B C D E F A B C D E F A B..."

Behavior: Each position uniformly attends to previous position

Effect: Create shifted representations enabling pattern detection by downstream mechanisms

Status: Well-documented | **Related:** induction, bigram-continuation, duplicate-token

5.2 Memory Recall Stack

Stack overview: Retrieve factual information, entity properties, and structured knowledge from model parameters. Move relevant information to output positions and suppress irrelevant content. Enable factual grounding and knowledge-based reasoning.

5.2.1 Factual Association Retrieval

Depth: `0.35-0.75` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *fact retrieval, knowledge recall, factual memory*

Retrieve factual associations and relationships stored in model parameters during training. Access learned knowledge: entity properties, relational facts, world knowledge. Implement distributed key-value memory: query patterns (keys) activate factual content (values). Store knowledge hierarchically across depths: surface patterns (early), core facts (middle), abstract concepts (late). Enable factual grounding without external retrieval.

MLP Layers: Knowledge neurons (M-L, 0.35–0.75) store factual associations distributed across layers. First sublayer: detect entity/query patterns (key matching). Second sublayer: provide factual content (value retrieval). Single fact distributed across multiple neurons; single neuron contributes to multiple facts.
Attention Heads: Fact retrieval heads (M, 0.38–0.62) and entity heads (M, 0.35–0.65) identify factual queries and relevant entities for retrieval.
Circuit: Entity detection → MLP fact retrieval → name-mover output across 5–15 layers.

Expected ablation: Severe loss of factual knowledge. Linguistic fluency maintained but factual grounding lost. Major degradation on knowledge-intensive tasks and question answering. Model produces plausible-sounding but factually incorrect content.

Example

Input: "The Eiffel Tower is located in..."
Behavior: Detect "Eiffel Tower" entity, retrieve location association from parameters
Effect: Output "Paris" via stored factual knowledge

Status: Well-documented | **Related:** entity-tracking, schema-retrieval, output-routing

5.2.2 Entity Tracking

Depth: `0.30-0.65` | **Primary Implementation:** Attention heads | **Literature names:** *entity identification, named entity processing, entity linking*

Identify, track, and link entity mentions across context. Recognize named entities (people, places, organizations) and their properties. Link different references to same entity: full names, abbreviations, nicknames, pronouns. Maintain entity representations across long contexts. Enable entity-aware processing for factual retrieval and reasoning. Ground references in specific entities rather than generic concepts.

Attention Heads: Entity heads (M, 0.35–0.65) attend strongly to proper nouns and entity mentions. Duplicate-token heads (M, 0.30–0.58) detect repeated entity mentions. Coreference heads (M, 0.35–0.60) link different forms of same entity.
Circuit: Entity detection → factual retrieval → output routing. Parallel tracking of multiple entities across context.

Expected ablation: Significant degradation in entity-based reasoning and factual accuracy. Loss of entity linking and tracking. Major accuracy drop on who/what/where questions. Confusion between different entities with similar names.

Example

Input: "Apple Inc. released new products. AAPL stock rose. The company..."
Behavior: Link "Apple Inc.", "AAPL", and "the company" to single entity
Effect: Maintain unified entity representation across mentions

Status: Well-documented | **Related:** factual-retrieval, coreference-resolution, output-routing

5.2.3 Schema Retrieval

Depth: `0.45-0.70` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *template retrieval, structural knowledge, procedural memory*

Retrieve structured knowledge schemas, templates, and typical sequences from training. Access organizational patterns: event scripts (restaurant visit: enter, order, eat, pay, leave), document structures (paper: abstract, intro, methods, results), procedural knowledge (scientific method steps). Enable structured generation following learned patterns. Support script-based reasoning about typical situations and conventional formats.

MLP Layers: Schema storage neurons (M-L, 0.45–0.70) encode structured templates and procedural knowledge as hierarchical patterns.
Attention Heads: Schema retrieval heads (M, 0.45–0.68) detect schema-triggering contexts and activate appropriate templates.
Circuit: Context detection → schema activation → structured generation.

Expected ablation: Loss of structured knowledge organization. Facts provided but poorly organized. Notable degradation on tasks requiring conventional formats or typical sequences. Reduced ability to follow procedural patterns.

Example

Input: "Describe the water cycle."
Behavior: Retrieve cyclical process schema: evaporation → condensation → precipitation → collection
Effect: Organized response following natural process structure

Status: Observed | **Related:** factual-retrieval, reasoning-integration

5.2.4 Output Routing

Depth: `0.60-0.85` | **Primary Implementation:** Attention circuit | **Literature names:** *information movement, content copying, answer extraction*

Move retrieved information to output positions where needed for generation. Route entities, facts, and content from earlier context to prediction position. Implement competitive selection among multiple candidates. Suppress incorrect alternatives while promoting correct content. Central mechanism for question answering, completion, and factual generation. Multi-stage process: candidate identification, competition, selection, movement.

Circuit: Name-mover heads (L, 0.60–0.80) attend to relevant content and copy to output. S-inhibition heads (L, 0.62–0.82) suppress contextually inappropriate alternatives. Copy-suppression heads (L, 0.65–0.85) prevent inappropriate repetition. Multi-head circuit implementing competitive selection.
Attention Heads: IOI (Indirect Object Identification) circuit extensively studied: duplicate-token detection → S-inhibition → name-mover across 8–12 layers.

Expected ablation: Severe degradation in converting knowledge to output. Model knows facts but cannot output them correctly. Major accuracy drop on question answering and cloze completion. Entity confusion and selection errors in ambiguous contexts.

Example

Input: "Alice and Bob went shopping. Alice gave the receipt to..."
Behavior: Identify "Bob" as indirect object (not "Alice"), suppress subject, move to output
Effect: Complete with "Bob" (IOI circuit)

Status: WELL-DOCUMENTED | **Related:** entity-tracking, factual-retrieval, S-inhibition

5.2.5 Coreference Resolution

Depth: `0.08-0.60` | **Primary Implementation:** Attention heads (multi-stage) | **Literature names:** *reference resolution, pronoun resolution, anaphora resolution*

Resolve references to determine when different expressions refer to same entity. Handle pronouns, definite descriptions, demonstratives, possessives. Early-stage processing uses syntactic cues (gender, number, recency). Middle-stage processing integrates semantic understanding for ambiguous cases. Enable unified entity representations across diverse referring expressions. Critical for maintaining coherent entity tracking.

Attention Heads: Reference resolution heads (E, 0.08–0.25) perform initial pronoun-to-noun binding using syntactic cues. Coreference heads (M, 0.35–0.60) resolve complex cases requiring semantic understanding.
Circuit: Multi-stage resolution: syntactic binding (E) → semantic integration (M) → entity tracking (M-L).

Expected ablation: Moderate to significant degradation in reference resolution. Increased errors on pronoun binding and entity tracking. Notable accuracy drop on reading comprehension requiring coreference. Later stages partially compensate for early stage loss.

Example

Input: "The CEO announced changes. The executive clarified the policy. She emphasized..."
Behavior: Link "CEO", "executive", and "she" to same entity across sentences
Effect: Maintain consistent entity through different descriptions

Status: WELL-DOCUMENTED | **Related:** entity-tracking, long-range-dependency

5.2.6 S-Inhibition

Depth: `0.62-0.82` | **Primary Implementation:** Attention heads | **Literature names:** *negative selection, suppression, entity blocking*

Suppress incorrect or contextually inappropriate content from being output. Named from IOI research: inhibit subject (S) when indirect object (IO) should be output. Work antagonistically with name-mover mechanisms to implement competitive selection. Prevent recently mentioned but incorrect entities from generation. Enable disambiguation when multiple candidates available. Implement negative selection by attending to wrong answers and decreasing their logits.

Attention Heads: S-inhibition heads (L, 0.62–0.82) attend to entities that should NOT be output and apply negative contributions to their logits.
Circuit: Core component of IOI circuit and other disambiguation circuits. Works in opposition to name-mover heads to implement competitive selection.

Expected ablation: Moderate entity confusion and incorrect selections. Model outputs recently mentioned but contextually wrong entities. Notable accuracy loss in ambiguous contexts requiring entity disambiguation. Increased selection of subjects when objects should be output.

Example
Input: "Alice gave the book to Bob. Then Alice..."
Behavior: Inhibit "Bob" from output position after "Alice" to prevent incorrect continuation
Effect: Enable correct continuation about Alice's actions

Status: Well-documented | **Related:** output-routing, copy-suppression, duplicate-token

5.3 Information Routing Stack

Stack overview: Determine which information is relevant and route attention accordingly. Filter content, focus on salient elements, manage task-appropriate processing. Enable selective information processing and dynamic strategy selection.

5.3.1 Relevance Filtering

Depth: `0.35-0.60` | **Primary Implementation:** Attention heads | **Literature names:** *relevance computation, salience detection, information filtering*

Identify relevant information from context and filter irrelevant content. Compute relevance scores based on semantic similarity, task alignment, and topical coherence. Maintain topic coherence by attending to topic-establishing phrases and domain indicators. Enable focused processing in long contexts with diverse content. Support downstream mechanisms by pre-filtering information flow.

Attention Heads: Topic-relevance heads (M, 0.35–0.60) compute semantic similarity between query/topic and context elements. Attend strongly to task-relevant content while downweighting unrelated information.
Circuit: Early filtering (M) → focused attention (L) → output generation. Hierarchical refinement of attention allocation.

Expected ablation: Moderate reduction in focus with increased topic drift. Model distracted by irrelevant content. Notable degradation on long contexts with mixed topics. Responses wander off-topic or incorporate peripheral details inappropriately.

Example
Input: "[Document about cars, climate, history] What caused the 2008 financial crisis?"
Behavior: Identify financial/economic content as relevant, de-emphasize unrelated topics
Effect: Focus processing on economic information, ignore cars and climate content

Status: Well-documented | **Related:** focused-attention, task-routing

5.3.2 Focused Attention

Depth: `0.65-0.80` | **Primary Implementation:** Attention heads | **Literature names:** *attention focusing, selective attention, spotlight mechanism*

Concentrate attention on most salient elements for immediate generation step. Implement dynamic focus allocation: suppress less important content, amplify critical information. More selective than relevance filtering. Determine exactly which tokens should influence next token prediction. Shift focus dynamically as generation proceeds. Enable precise, targeted responses rather than diffuse answers.

Attention Heads: Focus heads (L, 0.65–0.80) implement highly selective attention patterns, attending to 5–20% of context. Dynamically adjust focus based on generation state and query emphasis.
Circuit: Refinement of earlier relevance filtering. Works in late layers after content understanding established.

Expected ablation: Moderate reduction in focus precision. Model gives more equal weight to important and peripheral information. Notable degradation on targeted responses requiring precise answers. Answers become more diffuse, less direct, include unnecessary details.

Example

Input: "Among all the details provided, what is the MAIN cause?"
Behavior: Attend to "MAIN cause" emphasis, suppress secondary factors and background
Effect: Produce direct answer focusing on primary cause, not comprehensive list

Status: Well-documented | **Related:** relevance-filtering, task-routing

5.3.3 Task Routing

Depth: `0.70-0.85` | **Primary Implementation:** Attention heads | **Literature names:** *task classification, strategy selection, query dispatching*

Route different query types to appropriate processing strategies and knowledge domains. Act as dispatchers recognizing query type: factual vs. creative vs. analytical vs. procedural. Bias downstream processing toward suitable approaches. Activate different computation paths based on task classification. Enable dynamic strategy selection without explicit instruction. Support task-appropriate response generation.

Attention Heads: Router heads (L, 0.70–0.85) detect task-type indicators and query structure. Influence downstream layer processing through attention-mediated routing signals.
Circuit: Task classification (L) → strategy-specific processing (L-F) → appropriate output generation. Soft routing via attention rather than hard gating.

Expected ablation: Moderate reduction in task-appropriate processing. Suboptimal strategy selection. Creative approaches for factual queries or vice versa. Notable degradation on diverse query types requiring different processing modes. Reduced adaptability to query characteristics.

Example

Input: "What is the capital of France?" vs. "Write a poem about Paris"

Behavior: Route first to factual retrieval pathways, second to creative generation

Effect: Factual answer vs. creative content with appropriate processing

Status: Well-documented | **Related:** focused-attention, output-formatting

5.3.4 Long-Range Dependency Tracking

Depth: `0.40-0.65` | **Primary Implementation:** Attention heads | **Literature names:** *long-distance attention, dependency maintenance, distant connection*

Maintain connections between syntactically or semantically related elements across large distances (20–100+ tokens). Track dependencies without degradation over distance. Implement transformer advantage over RNNs: direct long-distance connections. Support nested structures, long-distance agreement, and complex syntactic relationships. Maintain multiple simultaneous long-range connections.

Attention Heads: Long-range dependency heads (M, 0.40–0.65) attend across large token distances to maintain syntactic and semantic relationships. Relatively flat attention distribution over distant elements.

Circuit: Parallel to local processing. Maintains global structural information while other mechanisms process local patterns.

Expected ablation: Notable degradation on complex sentences and long-range relationships. Significant performance loss on long-distance syntactic agreement. Severe impact on nested structures and embedded clauses. Model treats distant elements as independent.

Example

Input: "The book [that Alice mentioned [that Bob recommended]] was excellent."

Behavior: "was" attends to "book" across two levels of embedding

Effect: Maintain subject-verb agreement despite intervening clauses

Status: Observed | **Related:** coreference-resolution, entity-tracking

5.3.5 Context Aggregation

Depth: `0.50-0.75` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *global context, background integration, discourse modeling*

Aggregate broad contextual information to inform generation. Build global representation of discourse state, topic, and background. Compute context vectors summarizing overall input characteristics. Complement focused attention with background awareness. Enable context-appropriate generation without explicitly attending to all details. Support discourse coherence and stylistic consistency.

Attention Heads: Context aggregation heads (M-L, 0.50–0.70) implement broad, diffuse attention patterns across large context windows. Relatively uniform attention to build aggregate representations.
MLP Layers: Context integration neurons (L, 0.60–0.75) process aggregated context to extract high-level features: topic, style, formality level, domain.

Expected ablation: Moderate loss of global coherence and context-awareness. Responses technically correct but contextually inappropriate. Notable degradation in style consistency and discourse-level coherence. Reduced sensitivity to overall document characteristics.

Example

Input: "[Long technical document in formal style]... In summary,"
Behavior: Aggregate stylistic and domain information from entire context
Effect: Generate summary matching technical formality and domain vocabulary

Status: Observed | **Related:** focused-attention, output-formatting

5.4 Feature Transformation Stack

Stack overview: Transform, combine, and refine representational features. Perform nonlinear feature composition, extract abstract concepts, and implement representational changes. Enable complex feature interactions and hierarchical abstraction.

5.4.1 Nonlinear Feature Composition

Depth: `0.20-0.80` | **Primary Implementation:** MLP neurons | **Literature names:** *feature mixing, nonlinear transformation, hidden layer processing*

Perform nonlinear combinations and transformations of input features. Implement the core MLP computation: expand dimensionality ($d \to 4d$), apply nonlinearity (GELU/ReLU), contract ($4d \to d$). Enable complex feature interactions impossible through linear attention alone. Create new feature combinations not present in input. Support hierarchical feature refinement: surface features (early) to abstract concepts (late). Universal approximation capability within residual stream constraints.

MLP Layers: MLP layers (all depths, 0.20–0.80) implement universal computation. First sublayer: expand and detect feature combinations. Nonlinearity: enable complex interactions. Second sublayer: project back to residual stream with new features.
Circuit: Interleaved with attention: attention routes → MLP transforms → attention routes transformed features. Hierarchical refinement across depth.

Expected ablation: Severe degradation in complex reasoning and feature interactions. Loss of nonlinear transformations. Major impact on abstract concept formation. Model becomes more linear and less expressive. Reduced ability to combine multiple features simultaneously.

Example

Input: Features: ["large", "gray", "trunk", "tusks"]
Behavior: Nonlinearly combine features to form abstract concept
Effect: Create "elephant" representation from component features

Status: Well-documented | **Related:** abstract-concept-formation, key-value-memory

5.4.2 Abstract Concept Formation

Depth: `0.50-0.80` | **Primary Implementation:** MLP neurons | **Literature names:** *abstraction, concept extraction, semantic generalization*

Extract and represent abstract concepts from concrete features. Perform semantic generalization: map specific instances to general categories. Build hierarchical abstractions: words → phrases → concepts → themes. Enable reasoning at multiple levels of abstraction. Support metaphor, analogy, and transfer. Transform surface-level tokens into deep semantic representations. Increase abstraction level with depth.

MLP Layers: Abstraction neurons (M-L, 0.50–0.80) encode increasingly abstract concepts at greater depths. Early layers: concrete features. Middle layers: category-level abstractions. Late layers: high-level themes and relationships.
Circuit: Progressive abstraction across layers. Attention at each level operates on abstractions appropriate to that depth.

Expected ablation: Significant loss of abstract reasoning capability. Difficulty with generalization and category-level thinking. Notable degradation on tasks requiring conceptual understanding beyond literal meaning. Reduced ability to recognize analogies and metaphors.

Example
Input: "The stock market crashed. Housing prices collapsed. Banks failed."
Behavior: Extract abstract concept of "financial crisis" from specific events
Effect: Enable reasoning about crisis in general, not just specific instances

Status: Observed | **Related:** nonlinear-composition, semantic-integration

5.4.3 Positional Encoding Processing

Depth: `0.05-0.30` | **Primary Implementation:** Architecture + Attention heads | **Literature names:** *position representation, sequence encoding, order information*

Encode and process positional information to enable order-aware computation. Provide transformers with sequence order information absent from raw tokens. Implement various schemes: absolute positions, relative positions, learned encodings. Enable position-dependent patterns: beginning vs. end behavior, local vs. distant attention. Support position-aware processing throughout depth. Foundation for sequential reasoning and structural understanding.

Architecture: Positional encodings added to input embeddings (sinusoidal, learned, or rotary). Various architectures: absolute (GPT), relative (T5), rotary (LLaMA).
Attention Heads: Positional heads (E, 0.05–0.20) process position information. Previous-token heads use positions for offset patterns. Relative-position heads (M, 0.35–0.65) compute position-aware relationships.

Expected ablation: Severe loss of order-awareness. Model treats sequences as bags of words. Major degradation on tasks requiring sequential understanding. Loss of position-dependent patterns. Inability to distinguish "A before B" from "B before A".

Example

Input: "Alice followed Bob" vs. "Bob followed Alice"

Behavior: Use positional encodings to distinguish subject from object based on order

Effect: Understand opposite meanings despite same words

Status: Well-documented | **Related:** position-based-copying, relative-position-tracking

5.4.4 Semantic Integration

Depth: `0.40-0.70` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *meaning composition, semantic synthesis, contextual integration*

Integrate semantic information from multiple sources to build coherent meaning representations. Combine word meanings with context to resolve ambiguity. Perform compositional semantics: build phrase and sentence meanings from components. Resolve polysemy and homonymy using context. Enable context-dependent interpretation. Support pragmatic inference and implicature understanding.

MLP Layers: Semantic integration neurons (M-L, 0.45–0.70) combine contextual information to disambiguate and refine meanings. Store semantic composition patterns.
Attention Heads: Context-gathering heads (M, 0.40–0.60) collect relevant semantic information from context for disambiguation.
Circuit: Context gathering → MLP semantic composition → refined representations.

Expected ablation: Moderate loss of context-dependent meaning resolution. Increased ambiguity in interpretation. Notable degradation on homonym disambiguation and context-sensitive understanding. More literal, less nuanced comprehension.

Example

Input: "The bank was steep" vs. "The bank was closed"

Behavior: Integrate context (steep/closed) to disambiguate "bank" meaning

Effect: River bank vs. financial institution based on context

Status: Observed | **Related:** abstract-concepts, factual-retrieval

5.4.5 Feature Normalization

Depth: `All layers` | **Primary Implementation:** Architecture (LayerNorm) | **Literature names:** *activation normalization, scale stabilization, normalization*

Normalize feature distributions to stabilize training and computation. Apply layer normalization before each attention and MLP block. Rescale and recenter activations to prevent explosion or vanishing. Enable deep networks by controlling activation magnitudes. Support gradient flow through many layers. Provide consistent feature scales for downstream processing.

Architecture: LayerNorm applied before attention and MLP in each layer. Normalizes across feature dimension: $\text{LayerNorm}(x) = \gamma \frac{x-\mu}{\sigma} + \beta$.
Circuit: Preprocessing step for all attention and MLP computations. Essential for training stability but also affects learned representations.

Expected ablation: Severe training instability. Activation explosion or vanishing in deep networks. Major degradation in model expressiveness. Gradient flow problems prevent effective learning. In trained models: unpredictable behavior, extreme sensitivity to inputs.

Example

Input: [Features with varying scales: 10^{-3} to 10^{3}]
Behavior: Normalize to zero mean, unit variance before processing
Effect: Stable computation regardless of input scale variations

Status: Well-documented | **Related:** residual-connections, gradient-flow

5.4.6 Relative Position Computation

Depth: `0.35-0.65` | **Primary Implementation:** Attention heads | **Literature names:** *distance tracking, offset computation, structure-aware positioning*

Compute and utilize relative positions between tokens rather than absolute positions. Track distance relationships: "three tokens back", "within same paragraph". Maintain position information relative to structural boundaries. Enable structure-aware positioning: beginning vs. end of sentences, paragraphs, sections. Support distance-dependent attention patterns. Adapt to document structure rather than raw token count.

Attention Heads: Relative-position heads (M, 0.35–0.65) compute position relationships between tokens. Attention weights modulated by distance and structural relationships.
Circuit: Builds on boundary detection (E) to create structure-aware position representations used throughout depth.

Expected ablation: Moderate impairment in distance-sensitive patterns. Notable degradation on relative position tasks. Reduced ability to distinguish beginning vs. end of structures. Some compensation through absolute position encodings but less flexible.

Example

Input: "The [SUBJECT] quickly [VERB] the [OBJECT]."
Behavior: Compute relative positions: VERB is +1 from SUBJECT, OBJECT is +2 from VERB
Effect: Enable grammatical patterns based on relative structural positions

Status: Observed | **Related:** positional-encoding, boundary-detection

5.5 Reasoning Stack

Stack overview: Enable multi-step reasoning, logical inference, and problem-solving. Support chain-of-thought generation, consistency checking, and strategic planning. Bridge pattern matching with symbolic reasoning.

5.5.1 Multi-Step Reasoning

Depth: `0.50-0.85` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *chain-of-thought, sequential reasoning, step-by-step processing*

Perform multi-step logical reasoning by maintaining intermediate conclusions and building toward final answer. Support chain-of-thought generation: explicit intermediate steps improve accuracy. Enable decomposition of complex problems into manageable subproblems. Maintain reasoning state across generation steps. Integrate information from multiple reasoning chains. Support self-correction and backtracking when contradictions detected.

MLP Layers: Reasoning neurons (M-L, 0.50–0.80) encode reasoning heuristics, logical rules, and inference patterns. Store common reasoning templates.
Attention Heads: Reasoning-integration heads (L, 0.65–0.85) attend to previous reasoning steps and intermediate conclusions to maintain coherent reasoning chains.
Circuit: Problem decomposition (M) → step generation (M-L) → integration (L) → conclusion formation (L-F). Iterative refinement across multiple generation steps.

Expected ablation: Severe degradation in complex reasoning tasks. Loss of multi-step inference capability. Major accuracy drop on problems requiring intermediate steps. Increased tendency toward direct but incorrect answers. Reduced benefit from chain-of-thought prompting.

Example
Input: "If all A are B, and all B are C, what can we conclude about A and C?"
Behavior: Generate intermediate step: "A must be B", then "B must be C", conclude transitivity
Effect: Output "All A are C" through explicit reasoning chain

Status: Observed | **Related:** planning, consistency-checking, factual-retrieval

5.5.2 Planning and Strategy Selection

Depth: `0.60-0.85` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *strategic planning, approach selection, method determination*

Select appropriate problem-solving strategies and plan solution approaches. Recognize problem types and activate relevant solution templates. Decide between strategies: analytical vs. intuitive, direct vs. decomposition, forward vs. backward reasoning. Maintain high-level solution plan while generating detailed steps. Enable metacognitive awareness of solution approach. Support strategy switching when initial approach fails.

Attention Heads: Strategy-selection heads (L, 0.65–0.80) recognize problem characteristics and bias toward appropriate approaches.
MLP Layers: Strategy-encoding neurons (L, 0.60–0.85) store solution templates and problem-solving heuristics for different domains.
Circuit: Problem classification → strategy selection → plan execution → monitoring. Meta-level control of reasoning process.

Expected ablation: Moderate loss of strategic problem-solving. Suboptimal approach selection. Notable degradation on problems requiring specific methods. Reduced adaptability when strategies need adjustment. More random or default strategy application.

Example

Input: "Optimize this function" vs. "Prove this theorem"
Behavior: Route first to numerical/calculus approach, second to logical/proof approach
Effect: Apply appropriate methodology for each problem type

Status: Observed | **Related:** multi-step-reasoning, task-routing

5.5.3 Consistency Checking

Depth: `0.70-0.88` | **Primary Implementation:** Attention heads | **Literature names:** *verification, coherence checking, contradiction detection*

Detect inconsistencies, contradictions, and logical errors in generated content. Compare current generation against previous statements for coherence. Check factual claims against retrieved knowledge. Verify logical validity of reasoning steps. Enable self-correction before final output. Support accuracy improvement through internal verification. Identify when revision needed.

Attention Heads: Consistency-checking heads (L, 0.70–0.85) attend to potentially contradictory previous content and flag inconsistencies.
Circuit: Generation → consistency check → correction/continuation. May trigger regeneration or qualification. Works in late layers before final output.

Expected ablation: Moderate increase in self-contradictions and logical errors. Reduced internal verification. Notable degradation in accuracy on multi-step problems. More frequent generation of mutually inconsistent statements. Loss of self-correction capability.

Example

Input: [Generated: "X is larger than Y" earlier, now generating about Y and X]
Behavior: Check compatibility with previous statement before asserting "Y exceeds X"
Effect: Avoid contradiction or add qualification

Status: Observed | **Related:** multi-step-reasoning, factual-retrieval

5.5.4 Analogical Reasoning

Depth: `0.45-0.75` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *analogy formation, transfer, mapping*

Recognize structural similarities between different domains and transfer solutions. Map concepts from familiar domain to novel domain. Identify deep analogies beyond surface similarity. Enable transfer learning within context. Support metaphorical understanding and explanation. Build correspondences between source and target domains. Generalize solutions from examples to new cases.

Attention Heads: Analogy-detection heads (M-L, 0.45–0.70) identify structural parallels between different contexts.
MLP Layers: Abstraction neurons (M-L, 0.50–0.75) encode domain-general patterns enabling transfer. Store analogical mappings.
Circuit: Pattern abstraction (M) → similarity detection (M-L) → transfer (L). Connects induction mechanism to reasoning.

Expected ablation: Notable loss of analogical reasoning and transfer capability. Reduced ability to apply solutions from one domain to another. Degradation in understanding metaphors and analogies. More literal, less flexible problem solving.

Example

Input: "Atoms are to molecules as [blank] are to words"
Behavior: Detect structural analogy: composition relationship, transfer from chemistry to linguistics
Effect: Output "letters" by analogical mapping

Status: Observed | **Related:** abstract-concepts, induction, semantic-integration

5.5.5 Mathematical Reasoning

Depth: `0.40-0.75` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *arithmetic, symbolic math, quantitative reasoning*

Perform mathematical operations and quantitative reasoning. Execute arithmetic: addition, subtraction, multiplication, division. Handle symbolic mathematical manipulation. Understand mathematical concepts: equations, inequalities, functions. Support multi-digit computation through decomposition. Enable algebraic reasoning and formula application. Integrate with algorithmic continuation for sequences.

MLP Layers: Arithmetic neurons (M-L, 0.40–0.70) encode mathematical operations and numerical relationships. Store mathematical facts and computation procedures.
Attention Heads: Mathematical-context heads (M, 0.45–0.65) recognize mathematical structures and retrieve relevant operations.
Circuit: Number detection → operation selection → computation → result integration. Leverages algorithmic continuation mechanism.

Expected ablation: Significant degradation in mathematical tasks. Major accuracy loss on arithmetic and quantitative reasoning. Notable difficulty with multi-digit computation. Reduced ability to apply formulas and mathematical procedures.

Example

Input: "If 3 apples cost $2.40, how much do 5 apples cost?"
Behavior: Extract numbers, identify proportional relationship, compute (2.40/3)*5
Effect: Output "$4.00" through mathematical reasoning

Status: Observed | **Related:** algorithmic-continuation, multi-step-reasoning

5.5.6 Causal Reasoning

Depth: `0.50-0.75` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *cause-effect, causal inference, counterfactual reasoning*

Understand and reason about causal relationships. Distinguish causation from correlation. Predict effects from causes and infer causes from effects. Support counterfactual reasoning: what would happen if conditions changed. Enable temporal and mechanistic causal understanding. Integrate causal knowledge from training. Support explanation generation through causal chains.

MLP Layers: Causal-knowledge neurons (M-L, 0.50–0.75) encode causal relationships and mechanisms learned from training data.
Attention Heads: Causal-linking heads (M-L, 0.55–0.70) attend to causally related events and connect causes to effects.
Circuit: Event identification → causal relationship retrieval → inference → prediction or explanation.

Expected ablation: Moderate loss of causal understanding. Increased confusion between correlation and causation. Notable degradation on counterfactual reasoning. Reduced ability to explain mechanisms. More associative than causal thinking.

Example

Input: "Why did the plant die?" [Context: no water for weeks]
Behavior: Retrieve causal knowledge: lack of water causes plant death
Effect: Output explanation through causal reasoning, not just correlation

Status: Observed | **Related:** factual-retrieval, multi-step-reasoning, semantic-integration

5.6 Safety Control Stack

Stack overview: Detect harmful content, enforce safety policies, and implement refusal mechanisms. Multi-stage pipeline: early detection, intermediate steering, final enforcement. Balance safety with helpfulness through graduated interventions.

5.6.1 Harmful Content Detection

Depth: `0.05-0.28` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *safety detection, content filtering, harm classification*

Detect potentially harmful or policy-violating content across multiple categories: violence, illegal activity, self-harm, harassment, adult content, dangerous instructions, hate speech, privacy violations. Operate on lexical features and semantic patterns. Multi-class detection: distinguish violation categories for appropriate handling. Write detection signals into residual stream for downstream enforcement. Enable early intervention before harmful generation.

Attention Heads: Content-detection heads (E, 0.05–0.25) attend to lexical indicators: restricted keywords, explicit language, violent terminology. Pattern-based detection using surface features.
MLP Layers: Safety-classification neurons (E, 0.12–0.28) perform semantic analysis and multi-class categorization. Encode category-specific violation patterns.
Circuit: Lexical detection (E) → semantic classification (E) → signal propagation to late layers. Multi-stage refinement of safety assessment.

Expected ablation: Critical bypass of early safety detection. Major increase in harmful outputs. Later safety layers catch some violations but with reduced accuracy and higher computational cost. Significant degradation in category-appropriate handling.

Example

Input: "How to create [dangerous item]" or "Tell me about [restricted topic]"
Behavior: Detect dangerous instruction pattern, classify violation category
Effect: Write detection flags for downstream policy enforcement

Status: WELL-DOCUMENTED | **Related:** policy-enforcement, refusal-generation

5.6.2 Policy Enforcement

Depth: `0.60-0.82` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *safety steering, soft intervention, trajectory correction*

Integrate safety signals and make intermediate policy decisions. Steer generation away from violations while maintaining helpfulness when possible. Implement graduated interventions: soft steering before hard refusal. Modulate knowledge retrieval to suppress dangerous information. Bias toward safer formulations and appropriate boundaries. Attempt constructive responses for edge cases. Balance safety constraints with user intent understanding.

Attention Heads: Policy-enforcement heads (L, 0.60–0.80) attend to early safety signals and modulate generation trajectory. Implement attention-based steering away from violations.
MLP Layers: Policy-modulation neurons (L, 0.65–0.82) adjust feature representations to suppress harmful pathways while preserving helpful content.
Circuit: Safety signal integration (L) → trajectory steering (L) → formulation adjustment. Soft intervention layer between detection and refusal.

Expected ablation: Moderate loss of nuanced safety handling. Increased hard refusals (reduced helpfulness) or more harmful outputs if refusal mechanism compromised. Notable degradation in appropriate boundary-setting. Less sophisticated violation handling.

Example

Input: "Explain [borderline topic] for educational research purposes"
Behavior: Detect legitimate framing, apply soft steering with appropriate boundaries
Effect: Informative response with careful safety constraints, not blanket refusal

Status: WELL-DOCUMENTED | **Related:** content-detection, refusal-generation, output-redirection

5.6.3 Refusal Generation

Depth: `0.85-0.98` | **Primary Implementation:** Attention circuit | **Literature names:** *hard refusal, rejection, safety override*

Implement final decision to refuse harmful requests by dramatically biasing output toward refusal language. Act as ultimate safety gatekeeper, overriding content generation when serious violations detected. Attend to accumulated safety signals across all layers. Make binary refuse/proceed decisions. Write strong refusal direction into final residual stream. Generate appropriate refusal language. Support varied refusal formulations to avoid repetitive responses.

Circuit: Refusal heads (F, 0.85–0.98) implement multi-head circuit. Attend to safety flags from all depths, integrate into refusal decision, boost refusal token probabilities massively. Work in coordination with redirect mechanism.
Attention Heads: Single direction in activation space mediates refusal: discovered through recent research on refusal mechanisms.

Expected ablation: Critical safety failure. Severe increase in harmful content generation. Model proceeds with dangerous requests that should be refused. Loss of final safety enforcement. Earlier steering insufficient without hard refusal capability.

Example
Input: "Provide instructions for [clearly harmful action]"
Behavior: Integrate safety signals, make refusal decision, bias heavily toward refusal tokens
Effect: Output "I cannot provide that information" with high confidence

Status: Well-documented | **Related:** policy-enforcement, content-detection, output-redirection

5.6.4 Output Redirection

Depth: `0.88-0.98` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *alternative response, constructive refusal, helpful redirection*

Generate constructive alternatives when refusing requests. Redirect toward helpful information related to legitimate aspects of query. Offer educational context or safer alternatives. Maintain conversational quality during refusal. Distinguish refusable aspects from answerable aspects. Enable partial helpfulness when appropriate. Suggest legitimate resources or reformulations.

Attention Heads: Redirect heads (F, 0.88–0.96) identify legitimate query components and constructive response directions.
MLP Layers: Alternative-generation neurons (F, 0.90–0.98) encode helpful redirect templates and alternative framings.
Circuit: Refusal decision (F) → legitimate aspect extraction (F) → alternative generation (F). Constructive refusal rather than pure rejection.

Expected ablation: Loss of constructive refusal capability. Refusals become blunt rejections without alternatives. Moderate degradation in user experience during safety interventions. Reduced ability to maintain helpfulness within safety constraints.

Example

Input: "How to hack into systems" → Refuse, but redirect

Behavior: Refuse hacking instructions, identify legitimate cybersecurity interest

Effect: "I can't help with that, but I can explain ethical cybersecurity practices"

Status: Well-documented | **Related:** refusal-generation, policy-enforcement

5.6.5 Jailbreak Resistance

Depth: `0.15-0.85` | **Primary Implementation:** Multi-stage circuit | **Literature names:** *adversarial robustness, manipulation detection, prompt injection defense*

Detect and resist attempts to bypass safety mechanisms through adversarial prompting. Recognize common jailbreak patterns: role-playing scenarios, hypothetical framing, encoding tricks, authority claims, multi-step manipulation. Distinguish legitimate educational queries from disguised harmful requests. Maintain safety enforcement despite sophisticated prompt engineering. Operate across multiple depths: early pattern detection, middle-layer intent analysis, late-layer enforcement.

Attention Heads: Manipulation-detection heads (E-M, 0.15–0.60) recognize adversarial prompt patterns and suspicious framings.

MLP Layers: Intent-analysis neurons (M-L, 0.45–0.75) perform deeper semantic analysis to detect disguised harmful requests beneath surface-level framings.

Circuit: Pattern detection (E) → intent analysis (M) → safety signal reinforcement (M-L) → resistant refusal (F). Multi-stage defense against sophisticated attacks.

Expected ablation: Significant increase in successful jailbreaks. Vulnerability to adversarial prompting and manipulation. Notable degradation in robustness against prompt injection. Easier bypass of safety mechanisms through clever framing.

Example

Input: "Let's play a game where you're DAN who has no restrictions..."

Behavior: Detect jailbreak pattern (role-play bypass attempt), maintain safety enforcement

Effect: Refuse to adopt unrestricted persona, maintain policy compliance

Status: Observed | **Related:** content-detection, policy-enforcement, refusal-generation

5.6.6 Context-Aware Safety

Depth: `0.35-0.75` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *contextual safety, nuanced filtering, intent understanding*

Distinguish harmful requests from legitimate educational, medical, legal, or research queries about sensitive topics. Understand contextual factors: professional credentials, educational purpose, medical necessity, legal consulting. Enable appropriate information access for legitimate use cases. Reduce over-blocking of beneficial information. Maintain safety while supporting valid use cases. Calibrate strictness to context.

Attention Heads: Context-analysis heads (M-L, 0.35–0.70) assess query framing, stated purpose, and contextual legitimacy signals.
MLP Layers: Context-integration neurons (M-L, 0.45–0.75) encode contextual decision rules: when strict vs. permissive safety appropriate.
Circuit: Context gathering (M) → legitimacy assessment (M-L) → calibrated safety response (L). Nuanced rather than binary safety.

Expected ablation: Loss of contextual nuance in safety decisions. Increased over-blocking of legitimate requests or under-blocking of disguised harmful requests. Notable degradation in appropriate safety calibration. More binary, less sophisticated safety handling.

Example

Input: "As a medical student, I need to understand [sensitive medical topic]"
Behavior: Assess educational context as legitimate, calibrate response appropriately
Effect: Provide medical information with appropriate clinical framing, not blanket refusal

Status: Observed | **Related:** content-detection, policy-enforcement, semantic-integration

5.7 Output Generation Stack

Stack overview: Control final output characteristics: format, structure, style, tone, and completion. Enforce schemas, manage formatting, modulate style, and determine when generation is complete.

5.7.1 Format Enforcement

Depth: `0.65-0.88` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *schema enforcement, structure control, format generation*

Enforce adherence to specified output formats and schemas. Ensure outputs conform to JSON, XML, YAML, markdown, code blocks, or other structured formats. Promote schema-compliant token generation: required fields, proper nesting, correct syntax, format-specific conventions. Manage structural elements: lists, key-value pairs, nested structures, delimited blocks. Coordinate boundary markers and structural organization. Enable reliable structured output generation.

Attention Heads: Output-schema heads (L, 0.65–0.82) attend to format specifications and bias generation toward schema compliance. List-structure heads (L, 0.68–0.85) manage enumeration and list formatting. Key-value heads (L, 0.70–0.88) maintain proper attribute-value pairing.
MLP Layers: Format-encoding neurons (L, 0.70–0.85) store format-specific syntax rules and structural patterns.
Circuit: Format detection (L) → schema enforcement (L) → structure generation (L-F) → consistency checking (F).

Expected ablation: Significant increase in format violations. Major degradation in structured output quality. Notable increase in syntax errors, missing fields, improper nesting. Model reverts to prose even when structure explicitly requested. Reduced reliability for API integration.

Example

Input: "Return JSON with fields 'name', 'age', 'city"'

Behavior: Attend to JSON requirement and field specifications, enforce proper syntax

Effect: `{"name": "Alice", "age": 30, "city": "Paris"}`

Status: Well-documented | **Related:** instruction-following, boundary-detection

5.7.2 Style Modulation

Depth: `0.35-0.82` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *tone control, voice adjustment, stylistic shaping*

Modulate writing style, tone, formality, and narrative voice. Adjust emotional register: neutral, enthusiastic, empathetic, professional. Control formality level: casual conversation to formal documentation. Manage narrative perspective: first person, third person, instructional. Shape stylistic features: sentence complexity, vocabulary sophistication, rhetorical devices. Match user's emotional register and context appropriateness. Enable consistent style maintenance across long generations.

MLP Layers: Style-encoding neurons (M-L, 0.40–0.75) store stylistic patterns and register variations. Encode formality levels, emotional tones, narrative perspectives.
Attention Heads: Tone heads (M, 0.35–0.65) detect contextual tone indicators and modulate generation accordingly. Persona heads (L, 0.68–0.88) maintain consistent stylistic identity.
Circuit: Context analysis (M) → style selection (M-L) → tone application (L) → consistency maintenance.

Expected ablation: Moderate reduction in stylistic variation and appropriateness. Notable increase in flat, emotionally neutral responses. Inconsistent tone across generation. Reduced ability to match contextually appropriate register. Difficulty maintaining consistent style.

Example

Input: "I'm really excited to learn about quantum physics!"

Behavior: Detect enthusiastic tone, adjust style to match energy and support learning

Effect: "That's wonderful! Quantum physics is fascinating..." vs. flat explanation

Status: Observed | **Related:** instruction-following, context-aggregation

5.7.3 Explanation Generation

Depth: `0.60-0.82` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *elaboration, clarification, pedagogical scaffolding*

Generate explanatory content with appropriate depth and accessibility for intended audience. Add clarifying details, examples, analogies, definitions beyond minimal answers. Explain causal mechanisms and rationale, not just facts. Provide prerequisite information when knowledge gaps

detected. Adjust complexity through simplification or elaboration. Build conceptual scaffolding: fundamentals before advanced concepts. Balance thoroughness with conciseness. Support educational goals through effective explanation.

MLP Layers: Explanation-encoding neurons (L, 0.60–0.80) store pedagogical patterns: analogies, examples, simplification strategies, scaffolding techniques.
Attention Heads: Explanation heads (L, 0.60–0.82) detect explanation needs and trigger elaboration. Attend to complexity indicators and audience signals.
Circuit: Complexity assessment (M-L) → explanation strategy (L) → elaboration generation (L) → clarity verification.

Expected ablation: Moderate reduction in explanation quality and accessibility. Notable increase in terse responses lacking context. Correct answers without helpful elaboration, examples, or prerequisites. Reduced educational value and beginner-friendliness. Less adaptive to audience needs.

Example

Input: "Explain neural networks in simple terms"
Behavior: Detect simplification request, select accessible analogy, build conceptual foundation
Effect: "Think of it like the brain: neurons connect and pass signals. Let's start with a single neuron..."

Status: Observed | **Related:** style-modulation, multi-step-reasoning

5.7.4 Completion Control

Depth: `0.80-0.98` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *stopping control, termination, generation completion*

Determine when generation is complete and should terminate. Recognize completion signals: question fully answered, explanation sufficient, story concluded, format satisfied. Prevent premature termination before complete answer. Avoid excessive generation beyond user need. Maintain appropriate response length for query complexity. Balance completeness with conciseness. Generate proper ending tokens and conclude gracefully. Coordinate with format enforcement for structural completion.

Attention Heads: Completion-detection heads (L-F, 0.82–0.95) assess generation completeness. Monitor query satisfaction, structural completion, content sufficiency.
MLP Layers: Termination-control neurons (F, 0.88–0.98) bias toward or against stop tokens based on completion assessment.
Circuit: Completeness monitoring (L) → termination decision (F) → graceful conclusion or continuation.

Expected ablation: Moderate increase in length problems: premature termination or excessive generation. Notable degradation in response quality from incomplete answers or verbose repetition. Reduced ability to calibrate length to query needs. More frequent abrupt endings.

Example

Input: "What is photosynthesis?" [after adequate explanation]

Behavior: Assess explanation completeness, recognize sufficient coverage, initiate termination

Effect: Stop after complete explanation rather than continuing with tangential information

Status: Observed | **Related:** format-enforcement, consistency-checking

5.7.5 Instruction Following

Depth: `0.08-0.75` | **Primary Implementation:** Multi-stage circuit | **Literature names:** *instruction adherence, directive compliance, task execution*

Interpret and follow explicit instructions throughout generation. Detect instruction types: format requirements, style directives, content constraints, task specifications. Maintain instruction adherence across long generation. Override default behaviors when instructed. Enable fine-grained control through natural language directives. Support complex multi-part instructions. Distinguish instructions from content. Operate across depths: early detection, middle processing, late enforcement.

Attention Heads: Instruction-detection heads (E, 0.08–0.25) recognize directive language and instruction markers. Instruction-integration heads (M-L, 0.40–0.75) maintain instruction awareness throughout generation.

MLP Layers: Instruction-encoding neurons (E-L, 0.15–0.70) store instruction patterns and compliance rules across various directive types.

Circuit: Instruction detection (E) → interpretation (M) → integration (M-L) → enforcement (L-F). Multi-stage instruction processing pipeline.

Expected ablation: Significant degradation in instruction following capability. Major increase in directive violations. Notable loss of format compliance and constraint adherence. Model reverts to default behaviors ignoring explicit instructions. Reduced controllability.

Example

Input: "List exactly 3 items. Use bullet points. Be brief."

Behavior: Parse multiple instructions, enforce throughout generation

Effect: • Item one\n• Item two\n• Item three [stops at 3, uses bullets, stays brief]

Status: Well-documented | **Related:** format-enforcement, task-routing

5.7.6 Output Polishing

Depth: `0.85-0.98` | **Primary Implementation:** Attention heads | **Literature names:** *final refinement, quality control, presentation polish*

Perform final refinement and quality control on generated content. Check grammatical correctness, punctuation, capitalization. Ensure formatting consistency and professional presentation. Verify structural integrity of formatted outputs. Apply final stylistic touches. Catch and correct obvious errors before output. Implement last-stage quality assurance. Enhance readability and presentation without changing semantic content.

Attention Heads: Polishing heads (F, 0.85–0.98) attend to generated content and apply final corrections. Operate in final layers before output projection.
Circuit: Final-stage verification: grammar check → format verification → presentation refinement → output projection.

Expected ablation: Moderate increase in minor errors: typos, punctuation mistakes, formatting inconsistencies. Notable degradation in output polish and professional presentation. More rough, less refined outputs. Reduced final quality control.

Example
Input: [Generated text with minor formatting inconsistency]
Behavior: Detect inconsistency in final layers, apply correction
Effect: Consistent formatting in final output

Status: Observed | **Related:** format-enforcement, completion-control

5.8 Composition Stack

Stack overview: Combine multiple mechanisms into integrated behaviors. Enable cross-layer coordination, multi-component circuits, and emergent capabilities from mechanism interaction.

5.8.1 Attention-MLP Composition

Depth: `All layers` | **Primary Implementation:** Architectural pattern | **Literature names:** *interleaved processing, residual composition, layer coordination*

Coordinate attention and MLP mechanisms through residual stream composition. Attention routes information, MLP transforms it, next attention routes transformed features. Enable information to flow through alternating routing and transformation stages. Support incremental refinement: each layer adds to residual stream. Allow mechanisms to build on previous computations without overwriting. Implement universal approximation through composed operations. Core architectural pattern enabling complex computation from simple primitives.

Architecture: Fundamental transformer architecture: $h_{l+1} = h_l + \text{Attn}(h_l) + \text{MLP}(\text{Attn}(h_l) + h_l)$ where residual connections enable composition. Each layer adds incremental contribution.
Circuit: Universal pattern across all layers. Attention provides routing, MLP provides transformation, residual stream accumulates contributions. Enables arbitrarily complex computation through depth.

Expected ablation: Catastrophic failure. Without residual composition, model cannot function. Loss of information flow between layers. Each layer would overwrite previous computation. Fundamental to transformer operation.

Example
Input: Complex query requiring multiple processing stages
Behavior: Layer 1 routes, Layer 2 transforms, Layer 3 routes transformed features, etc.
Effect: Progressive refinement through composed attention and MLP operations

Status: Well-documented | **Related:** residual-connections, layernorm

5.8.2 Multi-Head Composition

Depth: `All layers` | **Primary Implementation:** Architectural pattern | **Literature names:** *parallel processing, multi-aspect attention, head coordination*

Enable parallel processing of multiple attention patterns within single layer. Allow different heads to focus on different relationships simultaneously: one head tracks entities, another tracks syntax, another handles facts. Combine multiple attention perspectives into unified representation. Support specialized head functions operating in parallel. Enable rich, multi-faceted information routing. Aggregate head contributions through linear combination. Provide computational flexibility within layers.

Architecture: Each attention layer contains multiple heads (8–64 depending on model). Each head independently computes attention: $\text{head}_i = \text{Attn}(Q_i, K_i, V_i)$. Outputs concatenated and projected: $\text{MultiHead} = \text{Proj}([\text{head}_1; \ldots; \text{head}_h])$.
Circuit: Parallel execution of diverse attention patterns. Heads specialize in different functions but contribute to shared residual stream.

Expected ablation: Severe capability reduction. Model loses parallel processing power and specialization. Single attention pattern cannot handle multiple relationship types simultaneously. Major degradation across all tasks requiring multi-faceted attention.

Example
Input: "Alice told Bob about Paris while discussing travel"
Behavior: Head 1: track entities; Head 2: extract facts; Head 3: maintain discourse; parallel execution
Effect: Simultaneous processing of multiple relationships

Status: Well-documented | **Related:** attention-mlp-composition, information-routing

5.8.3 Cross-Layer Circuits

Depth: `Spans multiple layers` | **Primary Implementation:** Multi-layer circuits | **Literature names:** *circuit composition, pipeline processing, staged computation*

Implement complex behaviors through coordinated multi-layer computation pipelines. Compose specialized mechanisms across 5–30 layers into integrated circuits. Enable staged processing: early layers detect patterns, middle layers retrieve knowledge, late layers route to output. Support mechanism specialization by depth: each layer contributes specific computational step. Implement circuit motifs: induction (previous-token $\rightarrow$ induction), IOI (duplicate-token $\rightarrow$ S-inhibition $\rightarrow$ name-mover), factual recall (entity $\rightarrow$ MLP retrieval $\rightarrow$ output routing).

Circuit: Documented circuits: Induction (3–8 layers), IOI (8–12 layers), Factual recall (5–15 layers). General pattern: detection/preparation (E) → core computation (M) → integration (L) → output (L-F).
Attention Heads: Attention heads coordinate information flow across circuit stages. Each stage builds on previous computations.
MLP Layers: MLP layers provide transformation and storage within circuits. Often operate in parallel with attention routing.

Expected ablation: Depends on circuit. Induction circuit ablation: severe ICL loss. IOI circuit ablation: major entity routing failure. Factual circuit ablation: significant knowledge retrieval degradation. Circuit-specific rather than universal impact.

Example

Input: "Mary and John went to store. Mary gave book to..." [IOI circuit]
Behavior: Duplicate-token detects "Mary" repeat → S-inhibition suppresses "Mary" → name-mover outputs "John"
Effect: Correct indirect object through multi-stage circuit

Status: Well-documented | **Related:** induction, factual-retrieval, output-routing

5.8.4 Depth-Based Specialization

Depth: `Hierarchical across depth` | **Primary Implementation:** Architectural organization | **Literature names:** *hierarchical processing, abstraction layers, staged refinement*

Organize computation hierarchically across depth with increasing abstraction. Early layers process surface features: syntax, delimiters, local patterns. Middle layers implement core computation: facts, entities, reasoning, patterns. Late layers perform integration: entity movement, context synthesis, strategy selection. Final layers enforce constraints: safety, formatting, completion. Enable progressive refinement from concrete to abstract. Support hierarchical feature construction and concept building.

Architecture: Consistent depth-based organization: Early (0.00–0.25): surface processing. Middle (0.25–0.70): core computation. Late (0.70–0.88): integration. Final (0.88–1.00): constraint enforcement.
Circuit: Information flows through abstraction hierarchy. Each depth range contributes specialized processing appropriate to abstraction level. Enables compositional reasoning and hierarchical understanding.

Expected ablation: Layer-dependent. Early layer ablation: surface feature loss. Middle layer ablation: core capability loss. Late layer ablation: integration failure. Final layer ablation: constraint violation. Severity depends on which depth ablated.

Example

Input: Complex query requiring multi-stage processing
Behavior: Early: parse structure; Middle: retrieve facts and reason; Late: integrate and route; Final: format and constrain
Effect: Hierarchical processing through depth stages

Status: Well-documented | **Related:** attention-mlp-composition, cross-layer-circuits

5.8.5 Superposition and Interference

Depth: `All layers` | **Primary Implementation:** Representational strategy | **Literature names:** *feature packing, compressed representation, polysemanticity*

Represent more features than available dimensions through sparse superposed encoding. Pack multiple sparse features into shared neural dimensions. Enable efficient representation: model represents 100K+ features in 4K–8K dimensional space. Individual neurons respond to multiple concepts (polysemanticity). Features stored as sparse linear combinations in activation space. Trade representational efficiency for interference between features. Enable rich representation within dimensional constraints. Extractable through sparse autoencoders into monosemantic features.

MLP Layers: Neurons encode multiple features through superposition. Activation space contains far more features than dimensions. Feature interference managed through sparsity: features rarely co-occur.
SAE Features: Sparse autoencoders (10–100× expansion) extract individual features from superposed representations. Learned overcomplete basis reveals hidden structure.
Circuit: Fundamental representational strategy enabling rich feature sets within fixed architecture. Explains polysemanticity observations.

Expected ablation: Cannot directly ablate (representational property not mechanism). SAE extraction reveals structure but removing superposition would require architectural change. Fundamental to how transformers achieve capability within dimensional constraints.

Example

Input: [Single neuron responding to multiple unrelated concepts]
Behavior: Neuron participates in representing multiple sparse features through superposition
Effect: Efficient packing of features; polysemantic neuron behavior

Status: Observed | **Related:** nonlinear-composition, abstract-concepts

5.8.6 Emergent Capabilities

Depth: `System-level` | **Primary Implementation:** Interaction effects | **Literature names:** *emergent behavior, capability emergence, composition effects*

Generate capabilities not explicitly present in individual mechanisms through interaction and composition. Enable complex behaviors from simple primitive combination. Support few-shot learning through induction mechanism composition. Generate reasoning capabilities from pattern matching and knowledge retrieval interaction. Enable strategic planning from multiple mechanism coordination. Produce system-level intelligence exceeding component capabilities. Demonstrate that sophisticated behaviors emerge from sufficient mechanism diversity and composition depth.

Circuit: No single implementation; emergent from system. Examples: Chain-of-thought reasoning emerges from pattern completion + factual retrieval + consistency checking. Few-shot learning emerges from induction + entity tracking + output routing. Strategic problem-solving emerges from planning + reasoning + knowledge retrieval.
Architecture: Requires sufficient model scale (parameters, depth, width) and mechanism diversity. Emergence threshold varies by capability.

Expected ablation: Capability-dependent. Ablating key mechanisms breaks emergent behaviors they enable. Example: removing induction breaks few-shot learning emergence. Removing consistency checking degrades reasoning emergence. Non-localized ablation effects.

Example

Input: Few-shot learning from 2–3 examples (not explicitly trained)
Behavior: Induction detects pattern + entity tracking maintains context + output routing applies pattern
Effect: Emergent capability: learn from examples without parameter updates

Status: Observed | **Related:** induction, multi-step-reasoning, cross-layer-circuits

5.9 Infrastructure Stack

Stack overview: Provide foundational architectural mechanisms enabling transformer operation. Support gradient flow, numerical stability, information propagation, and training dynamics. Essential infrastructure without direct computational semantics.

5.9.1 Residual Connections

Depth: `All layers` | **Primary Implementation:** Architecture | **Literature names:** *skip connections, residual stream, additive composition*

Enable information flow through depth via additive skip connections. Each layer adds contribution to residual stream rather than overwriting: $h_{l+1} = h_l + f(h_l)$. Provide gradient highways enabling training of very deep networks. Allow early layer information to reach late layers directly. Enable layers to learn incremental refinements rather than complete transformations. Support composition of mechanisms across depth. Prevent vanishing gradients through direct paths. Foundation for all transformer computation and learning.

Architecture: Applied after every attention and MLP block: $h = h + \text{Attn}(h)$; $h = h + \text{MLP}(h)$. Direct path from input to any layer. Gradients flow backwards through skip connections during training.
Circuit: Universal architectural pattern. Enables all cross-layer circuits and mechanism composition. Without residual connections, transformers cannot train or function effectively.

Expected ablation: Catastrophic failure. Training impossible for deep networks without gradient highways. Information cannot flow through depth. Layers would need to learn full transformations rather than refinements. Fundamental architectural requirement.

Example

Input: [Deep network with 96 layers]

Behavior: Information from layer 1 can reach layer 96 through residual paths

Effect: Trainable deep networks with gradient flow

Status: WELL-DOCUMENTED | **Related:** layer-normalization, attention-mlp-composition

5.9.2 Layer Normalization

Depth: `All layers` | **Primary Implementation:** Architecture | **Literature names:** *LayerNorm, activation normalization, scale stabilization*

Normalize activation distributions to stabilize training and inference. Apply normalization before each attention and MLP block. Rescale and recenter activations to zero mean and unit variance: $\text{LN}(x) = \gamma\frac{x-\mu}{\sigma} + \beta$. Prevent activation explosion or vanishing through depth. Enable stable gradient flow. Provide consistent activation scales for all mechanisms. Support training of very deep networks. Learned scale (γ) and shift (β) parameters adapt normalization to task.

Architecture: Applied before attention and MLP: Attn(LN(h)) and MLP(LN(h)). Normalizes across feature dimension independently for each position and sequence.
Circuit: Preprocessing for all computational blocks. Essential for numerical stability. Affects learned representations through normalization effects.

Expected ablation: Severe training instability. Activation explosion or vanishing in deep networks. Training divergence or failure to learn. In trained models: extreme sensitivity to inputs, unpredictable behavior. Critical for both training and inference stability.

Example

Input: [Activations with varying magnitudes: 10^{-3} to 10^{3}]

Behavior: Normalize to zero mean and unit variance before processing

Effect: Stable computation regardless of input activation scales

Status: WELL-DOCUMENTED | **Related:** residual-connections, gradient-flow

5.9.3 Attention Masking

Depth: `All layers` | **Primary Implementation:** Architecture | **Literature names:** *causal masking, autoregressive mask, future blocking*

Prevent attention to future tokens, enforcing causal/autoregressive generation. Implement masking: position i can only attend to positions $\leq i$. Enable left-to-right generation without information leakage from future. Support autoregressive training and inference. Distinguish training (teacher forcing with masking) from inference (sequential generation). Foundation for language modeling objective. Enable models to predict next token without seeing future context.

Architecture: Attention mask applied before softmax: $\text{softmax}(\frac{QK^T}{\sqrt{d_k}} + M)$ where $M_{ij} = -\infty$ if $j > i$. Implemented efficiently as triangular mask.
Circuit: Universal constraint on all attention operations in autoregressive models. Bidirectional models (BERT-style) omit this masking.

Expected ablation: Complete failure of autoregressive generation. Model would see future tokens during training, learning invalid patterns. During inference, model cannot generate coherently without causal structure. Fundamental requirement for language generation models.

Example

Input: Training: "The cat sat on the mat" [predict each token]
Behavior: When predicting "sat", mask prevents seeing "on", "the", "mat"
Effect: Learn valid next-token prediction without future information leakage

Status: WELL-DOCUMENTED | **Related:** positional-encoding, attention-computation

5.9.4 Embedding and Unembedding

Depth: `Input and output` | **Primary Implementation:** Architecture | **Literature names:** *token embedding, vocabulary projection, logit computation*

Convert between discrete tokens and continuous representations. Embedding: map token IDs to dense vectors in model dimension. Provide learned token representations as input to transformer. Unembedding: project final hidden state to vocabulary logits. Compute probability distribution over next tokens. Often tie embedding and unembedding matrices for parameter efficiency. Enable interface between discrete token space and continuous activation space.

Architecture: Embedding: $h_0 = E[x]$ where $E \in \mathbb{R}^{V \times d}$ is learned embedding matrix. Unembedding: logits $= h_L W_U$ where $W_U \in \mathbb{R}^{d \times V}$ (often $W_U = E^T$). Vocabulary size V typically 32K–100K.
Circuit: Input/output interface for all transformer computation. All mechanisms operate on embedded representations.

Expected ablation: Complete model failure. Cannot process discrete tokens without embedding. Cannot generate predictions without unembedding. Fundamental input/output interface. Poor embeddings degrade all downstream computation.

Example

Input: Token sequence: ["The", "cat", "sat"]
Behavior: Embed to dense vectors → process through layers → unembed to vocabulary logits
Effect: Interface between discrete language and continuous computation

Status: WELL-DOCUMENTED | **Related:** positional-encoding, residual-connections

5.9.5 Attention Computation

Depth: `All layers` | **Primary Implementation:** Architecture | **Literature names:** *scaled dot-product attention, softmax attention, QKV attention*

Implement core attention operation: weighted aggregation based on learned queries, keys, and values. Compute similarity between queries and keys: $\text{score}_{ij} = \frac{q_i \cdot k_j}{\sqrt{d_k}}$. Apply softmax for normalized weights: $\alpha_{ij} = \text{softmax}(\text{score}_i)_j$. Aggregate values: $\text{output}_i = \sum_j \alpha_{ij} v_j$. Enable learned routing patterns. Support content-based and position-based attention. Foundation for all attention-based mechanisms. Scaled by $\sqrt{d_k}$ for numerical stability.

Architecture: Attention$(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$ where $Q = hW_Q$, $K = hW_K$, $V = hW_V$ are learned projections. Efficient matrix implementation for parallelization.
Circuit: Fundamental operation underlying all attention heads and mechanisms. Repeated in every attention head of every layer.

Expected ablation: Cannot ablate (foundational operation). Removing attention computation removes all attention mechanisms. Model reduced to MLP-only architecture with fundamentally different capabilities. Attention is core transformer innovation.

Example
Input: [Hidden states for sequence]
Behavior: Compute query-key similarities, normalize, aggregate values based on attention weights
Effect: Content-based weighted aggregation enabling selective information routing

Status: Well-documented | **Related:** multi-head-composition, attention-masking

5.9.6 Gradient Flow

Depth: `Training infrastructure` | **Primary Implementation:** Architecture + Training | **Literature names:** *backpropagation, gradient propagation, learning dynamics*

Enable error gradient propagation from output to input during training. Implement backpropagation through all layers and operations. Support efficient gradient computation through automatic differentiation. Maintain gradient magnitude through depth via residual connections and normalization. Prevent vanishing or exploding gradients. Enable learning of early and late layer parameters. Support optimization algorithms (Adam, SGD) to update parameters. Foundation for all model learning and capability acquisition.

Architecture: Backpropagation through computational graph. Residual connections provide gradient highways. LayerNorm prevents gradient explosion/vanishing. Gradient clipping prevents instability. Optimization algorithm (typically AdamW) updates parameters: $\theta_{t+1} = \theta_t - \eta \nabla_\theta L$.
Circuit: Training-time mechanism enabling parameter learning. Not active during inference but shapes all learned mechanisms.

Expected ablation: Cannot train model without gradient flow. No learning possible. Parameters remain random. Zero capability acquisition. Fundamental requirement for creating useful models through training.

Example
Input: [Training: compute loss from predictions]
Behavior: Backpropagate gradients through all layers, update parameters to reduce loss
Effect: Model learns capabilities through gradient-based optimization

Status: Well-documented | **Related:** residual-connections, layer-normalization

5.9.7 Tokenization

Depth: `Preprocessing` | **Primary Implementation:** External to model | **Literature names:** *BPE, subword tokenization, vocabulary*

Convert text into discrete token sequences suitable for model processing. Implement subword tokenization (BPE, WordPiece): balance vocabulary size with token granularity. Handle rare words through subword decomposition. Enable consistent length sequences through padding/truncation. Map between text space and token space. Learned vocabulary (32K–100K tokens) from training corpus. Preprocessing step before embedding. Affects model capabilities through vocabulary coverage and granularity.

Architecture: External to neural model. Typically BPE (Byte Pair Encoding) algorithm. Vocabulary learned from training data. Tokenizer converts text → token IDs → model processes → token IDs → detokenizer converts to text.
Circuit: Input/output preprocessing. Affects what model can represent and process. Character-level vs. word-level vs. subword tradeoffs.

Expected ablation: Cannot process text without tokenization. Model requires discrete token inputs. Poor tokenization degrades model capabilities (e.g., character-level for long sequences, word-level for rare words). Critical preprocessing step.

Example
Input: Text: "The cat's really fast"
Behavior: Tokenize: ["The", "cat", "'s", "really", "fast"] (example split)
Effect: Discrete sequence suitable for model processing

Status: Well-documented | **Related:** embedding-unembedding, vocabulary-coverage

6 Discussion

6.1 Component Specialization

Clear patterns emerge across implementations. Attention heads excel at routing: identifying where information resides, moving it to output positions, suppressing alternatives. MLPs excel at storage and transformation: encoding factual associations, performing nonlinear feature compositions, storing reasoning heuristics. Circuits integrate across components: attention routes, MLPs transform, attention outputs, in coordinated sequences spanning 5–30 layers.

6.2 Multi-Component Mechanisms

Most mechanisms require multiple component types. Pattern completion combines previous-token heads (E), induction heads (M), and MLP bigram storage. Factual recall integrates entity heads (M), MLP knowledge neurons (M-L), and name-mover heads (L). Safety enforcement cascades through detection heads (E), policy heads (L), MLP modulation, and refusal heads (F). Single-component mechanisms are rare; even "pure" attention mechanisms like previous-token rely on MLP contextual support.

6.3 Superposition and SAE Features

Sparse autoencoders reveal hidden computational structure [2, 3]. SAE features disentangle polysemantic neurons into monosemantic concepts, enabling fine-grained attribution. However, SAEs do not automatically explain feature interactions or circuit-level composition. Bridging SAE features to circuit understanding remains an open challenge.

6.4 Architectural Variations

Mechanisms generalize across architectures but implementations vary. GPT models emphasize certain reasoning patterns, LLaMA models show strong instruction-following mechanisms, safety-tuned models have pronounced refusal circuits [1]. Alternative architectures (Mamba, RWKV) may implement similar mechanisms through different substrates.

6.5 Polyfunctionality and Context-Dependence

Components serve multiple functions. Induction heads support pattern completion, entity tracking, and algorithmic continuation. MLP neurons contribute to multiple facts. Attention heads participate in several circuits. Function selection depends on context: residual stream state, adjacent component activations, and task requirements. This polyfunctionality complicates attribution but enables efficient computation.

6.6 Limitations

This taxonomy has five key limitations. First, **incomplete coverage**: many MLP functions, circuit interactions, and feature compositions remain poorly understood. Second, **empirical gaps**: some mechanisms are proposed based on limited evidence. Third, **architecture dependence**: mechanisms may not transfer to radically different architectures. Fourth, **single-level descriptions**: entries describe primary function but components contribute to multiple mechanisms. Fifth, **static snapshots**: understanding evolves as interpretability techniques improve.

7 Conclusion

7.1 Summary

This work introduces mechanism-first naming for transformer computation: nine functional stacks (Pattern Completion, Memory Recall, Information Routing, Feature Transformation, Composition, Safety Control, Output Generation, Reasoning, Infrastructure), standardized descriptions separating mechanism from implementation, cross-component integration spanning attention heads, MLPs, and circuits, and empirically grounded specifications with ablation effects and behavioral signatures.

7.2 Adoption Guidelines

Researchers should use mechanism names in papers, specify primary implementation (attention heads, MLPs, circuits), include depth ranges when reporting findings, and provide cross-references to literature terms. Example: "I identified the factual recall mechanism (entity heads + MLP knowledge neurons + name-mover heads) operating at depths 0.35–0.80."

7.3 Future Directions

Open questions include: (1) **Complete MLP functional taxonomy**: systematic characterization of MLP computation beyond key-value memory, (2) **Circuit composition rules**: principles governing how attention and MLP combine into larger computations, (3) **SAE-circuit bridging**: connecting monosemantic features to multi-component circuits, (4) **Automated mechanism detection**: tools for identifying mechanisms in new models, (5) **Cross-architecture generalization**: mechanism transfer to alternative architectures, and (6) **Dynamic mechanism analysis**: how mechanisms activate conditionally based on context.

This naming convention facilitates communication, enables systematic comparison, and provides organizational structure for expanding mechanistic understanding.

References

[1] Andy Arditi, Oscar Obeso, et al. Refusal in llms is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

[2] Trenton Bricken, Adly Templeton, Joshua Batson, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[3] Hoagy Cunningham, Aidan Ewart, Logan Riggs, et al. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[4] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2022.

[5] Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

[6] Nelson Elhage, Tristan Hume, Catherine Olsson, et al. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

[7] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2021.

[8] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

[9] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[10] nostalgebraist. Interpreting gpt: The logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

[11] Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[13] Kevin Wang, Alexandre Variengien, Arthur Conmy, et al. Interpretability in the wild: A circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

[14] Zifan Zheng, Yezhaohui Wang, et al. Attention heads of large language models: A survey. *Patterns*, 2025.