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

Karol Kowalczyk

November 2025

Abstract

Transformer models implement computation through mechanisms that span multiple organizational levels: attention heads route information, MLP layers store and transform features, and multi-component circuits integrate these primitives into complex behaviors. Current mechanistic interpretability research suffers from fragmented naming—the same mechanism appears under different names, different mechanisms share identical names, and descriptions conflate computational function with implementation substrate.

This work proposes a mechanism-first taxonomy organizing transformer computation into eight functional stacks containing 35 abstract mechanisms. Each mechanism is described independently of implementation, then mapped to its constituent components (attention heads, MLPs, circuits, SAE features, or architectural elements). This framework enables cross-architecture comparison, supports multi-component integration, and provides stable vocabulary for the interpretability community. The taxonomy synthesizes findings from attention head analysis, MLP neuron studies, circuit tracing, and sparse autoencoder research into a unified mechanistic perspective.

Contents

1 Introduction

1.1 The Fragmented State of Mechanism Naming

Mechanistic interpretability has made remarkable progress reverse-engineering transformer computation. Researchers have identified induction heads that enable few-shot learning [10], characterized MLP layers as distributed key-value memories [6], traced multi-component circuits like the 26-head IOI circuit [12], and extracted monosemantic features from superposed representations [1]. Yet despite this empirical richness, naming conventions remain inconsistent and component-centric.

The same computational mechanism appears under multiple names depending on research tradition. Pattern completion is called "induction head" in circuit research, "pattern head" in attention studies, "copy head" in some papers, and "ICL head" when emphasizing in-context learning. Conversely, the term "copy head" refers to at least three distinct mechanisms: induction-based pattern completion, name-mover output routing, and position-based copying. This naming fragmentation hinders communication, prevents systematic comparison across models, and obscures the relationship between components and the mechanisms they implement.

The root problem is **component-first thinking**: organizing mechanisms by their implementation substrate (attention heads, MLP neurons, circuits) rather than their computational function. This creates artificial boundaries. Attention head taxonomies describe routing patterns but ignore MLP contributions. MLP neuron studies characterize feature storage but disconnect from attention-based retrieval. Circuit analyses trace information flow but lack standardized mechanism vocabulary. The field needs mechanism-first organization that describes *what is computed* independently, then specifies *how components implement it.*

1.2 A Mechanism-First Framework

This work introduces a mechanism-first taxonomy with three organizing principles:

1. Separate mechanism from implementation. Each mechanism entry describes the computational transformation in implementation-agnostic terms, then specifies which components (attention heads, MLPs, circuits, SAE features, architectural elements) realize it and how. Pattern completion is described as detecting and completing [A][B]...[A] $\rightarrow$ [B] patterns, then mapped to its multi-component implementation: previous-token heads create shifted representations, induction heads match patterns, MLP neurons store common n-grams.

2. Organize by computational function, not substrate. Mechanisms are grouped into eight functional stacks based on their role in transformer computation: Pattern & Sequential processing, Memory & Knowledge retrieval, Routing & Context management, Feature Transformation, Reasoning & Inference, Safety & Policy enforcement, Output & Quality control, and Composition & Integration. This organization reflects the computational pipeline rather than component boundaries.

3. Support multi-component integration. Most sophisticated mechanisms require coordinated computation across attention heads, MLP layers, and multiple depths. The taxonomy explicitly represents this composition: factual recall integrates entity heads (query identification), MLP knowledge neurons (fact storage), and name-mover heads (output routing) across 5–15 layers. Representing mechanisms as multi-component systems enables understanding of circuit-level behavior.

1.3 Scope and Approach

This taxonomy catalogs 35 abstract mechanisms organized across eight functional stacks, plus eight infrastructure primitives documented in appendix. Mechanisms are included when they meet four criteria: (1) independently observed across multiple models, (2) behaviorally validated through ablation studies, (3) mechanistically understood at component level, and (4) reproducible using standard interpretability techniques. Proposed mechanisms with limited empirical support are clearly marked.

The taxonomy acknowledges three fundamental limitations. First, **mechanisms are polyfunctional**—single components contribute to multiple computations depending on context. Second, **implementation varies across architectures**—while mechanisms generalize (most transformers perform pattern completion), specific implementations differ between GPT, LLaMA, and Claude families. Third, **understanding is incomplete**—many MLP functions, circuit interactions, and emergent behaviors remain poorly characterized.

Each mechanism entry provides: (1) functional description (computational transformation), (2) primary implementation (which component types), (3) implementation details (attention patterns, MLP structure, circuit composition), (4) depth distribution (where in the network), (5) ablation effects (what breaks when removed), and (6) concrete examples (input $\rightarrow$ mechanism $\rightarrow$ output).

1.4 Contributions

This work provides the interpretability community with:

1. **Standardized mechanism vocabulary** bridging attention head, MLP, circuit, and SAE research traditions through consistent naming independent of implementation details
2. **Functional stack organization** grouping 35 mechanisms by computational role rather than component type, revealing the transformer's computational architecture
3. **Multi-component integration** explicitly representing how attention heads, MLPs, and circuits compose into sophisticated mechanisms like factual recall and safety enforcement
4. **Implementation specifications** detailing how different component types realize each mechanism, enabling cross-architecture comparison and mechanism detection
5. **Empirically grounded descriptions** synthesizing findings from attention pattern analysis, MLP probing, circuit tracing, activation patching, and sparse autoencoder studies

1.5 Document Structure

Section 2 reviews transformer architecture and the three levels of mechanistic organization (heads, MLPs, circuits). Section 3 explains the transition from primitive components to abstract mechanisms and details mechanism identification methodology. Section 4 introduces the depth-based organization model spanning Early, Middle, Late, and Final layers.

Sections 6.1 through 6.8 catalog mechanisms by functional stack, providing complete descriptions with implementation details, ablation effects, and examples. Section 7 analyzes how mechanisms compose into circuits and produce emergent capabilities. Section 8 formalizes the mapping from abstract mechanisms to concrete components.

Section 9 examines cross-stack patterns, discusses limitations, and analyzes polyfunctionality. Section 10 provides adoption guidelines and identifies directions for future research.

2 Background and Related Work

2.1 Transformer Architecture

Transformers [11] process token sequences through L layers, each containing multi-head attention and feed-forward (MLP) sublayers operating on a shared residual stream. At layer l, computation proceeds: $h'_l = h_{l-1} + \text{Attn}(h_{l-1})$ followed by $h_l = h'_l + \text{MLP}(h'_l)$, where residual connections enable additive composition. Layer normalization stabilizes activations before each sublayer. The architecture implements three key principles: attention provides learned routing, MLPs provide nonlinear transformation, and residual connections enable information flow across depth.

Multi-head attention computes $\text{Attn}(h) = \sum_{i=1}^{H} W_O^i \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}}) V_i$ where $Q_i = hW_Q^i$, $K_i = hW_K^i$, $V_i = hW_V^i$ are learned projections. Each head attends to different positions independently, enabling parallel processing of diverse relationships. MLP layers expand dimensionality ($d \to 4d$), apply nonlinearity (GELU or ReLU), and contract ($4d \to d$): $\text{MLP}(h) = W_2\sigma(W_1 h)$. This two-layer structure enables universal approximation within the residual stream constraint.

2.2 Three Levels of Mechanistic Organization

Mechanistic interpretability research has characterized transformer computation at three organizational levels, each revealing different aspects of the computational structure.

Attention head analysis identifies specialized routing patterns [4, 10]. Previous-token heads attend uniformly to position $i-1$ from position i, creating shifted token representations. Induction heads implement content-based matching: from position i, attend to positions where the previous token matches the current token's predecessor, enabling pattern completion. Name-mover heads attend to entities mentioned earlier and copy them to output positions. Duplicate-token heads perform exact token matching across arbitrary distances. These patterns reveal attention's role as learned, conditional routing.

MLP neuron studies characterize feed-forward layers as distributed associative memories [6, 7]. The key-value memory interpretation views first-layer weights W_1 as detecting patterns (keys) and second-layer weights W_2 as providing associated content (values). Knowledge neurons encode specific facts: individual neurons activate for particular entities or relationships, and their activation causally influences factual predictions [3, 8]. MLP computation is distributed—single facts engage multiple neurons, single neurons contribute to multiple facts—but systematic patterns emerge across models.

Circuit tracing reveals multi-component computation spanning layers and sublayers [12]. The indirect object identification (IOI) circuit uses 26 attention heads across 8–12 layers: duplicate-token heads detect repeated names, S-inhibition heads suppress the repeated entity, and name-mover heads copy the alternative entity to output. Activation patching validates these circuits by showing that information flows through specific paths. Circuits demonstrate that sophisticated behaviors require coordinated attention and MLP computation, not individual components operating independently.

2.3 Superposition and Sparse Autoencoders

Neurons exhibit polysemanticity: individual neurons respond to multiple unrelated concepts, complicating interpretation [5]. This arises from **superposition**—models represent more features than available dimensions by encoding features as sparse linear combinations in activation space. Features interfere when they co-occur, but sparsity keeps interference bounded, enabling efficient representation within dimensional constraints.

Sparse autoencoders (SAEs) address polysemanticity by learning overcomplete sparse decompositions [1, 2]. An SAE with 10–100× expansion (e.g., 40K features from 4K dimensions) learns monosemantic features that activate for specific concepts. SAE features enable fine-grained attribution but do not automatically explain feature interactions or circuit-level composition. Bridging SAE features to mechanism understanding remains active research.

2.4 The Need for Unified Naming

Current research traditions use different vocabularies shaped by their methodological focus. Attention papers describe "induction heads" and "name-mover heads." MLP studies discuss "knowledge neurons" and "key-value memories." Circuit analyses reference components by layer and position ("L9H6") rather than function. This fragmentation prevents integration: a researcher studying factual recall must manually connect entity heads, MLP knowledge neurons, and name-mover circuits despite these components implementing a unified mechanism.

Inconsistent naming also hinders replication and comparison. When one paper's "pattern head" is another's "induction head" and a third's "ICL mechanism," determining whether findings replicate across models requires reverse-engineering terminology. Cross-architecture comparison becomes difficult when mechanisms are described through component-specific language rather than computational function.

This taxonomy addresses these problems by providing mechanism-first descriptions that abstract over implementation details, enabling researchers to communicate about computational functions while maintaining precision about how those functions are realized in different models.

3 Methodology: From Components to Mechanisms

3.1 The Abstraction Challenge

Transformer interpretability began with component-level analysis: identifying what individual attention heads do ("this head attends to the previous token"), characterizing individual MLP neurons ("this neuron activates for Paris"), or tracing specific circuits ("the IOI circuit uses these 26 heads"). While empirically grounded, this approach creates a vocabulary explosion—each component variant receives a name, mechanisms implemented differently across models appear unrelated, and the computational forest disappears behind component-level trees.

The challenge is moving from primitive components to abstract mechanisms without losing empirical grounding. An abstract mechanism must satisfy four requirements: (1) **implementation-independence**—describe the computational transformation without assuming specific components, (2) **empirical grounding**—map to observed components whose behavior validates the mechanism, (3) **cross-model generalization**—apply across architectures despite implementation variation, and (4) **compositional clarity**—specify how components combine when mechanisms require multi-component implementation.

3.2 Mechanism Identification Methodology

Mechanisms are identified through converging evidence from five complementary techniques:

Attention pattern analysis reveals specialized routing through visualization and statistical characterization. Previous-token heads show diagonal attention patterns (uniform weight at offset -1). Induction heads show characteristic stripes: high attention where previous tokens match current context. Name-mover heads attend strongly to specific entity positions. Quantitative metrics (attention entropy, pattern locality) enable systematic head classification [4].

Ablation studies assess behavioral necessity by removing components and measuring performance degradation. Mean ablation (replacing activations with layer-mean) tests whether components contribute causally. Zero ablation (removing components entirely) reveals strong dependencies. Task-specific metrics (few-shot accuracy, factual recall, entity disambiguation) quantify mechanism importance for different capabilities [12].

Activation patching traces information flow by restoring clean-run activations into corrupted runs. Patching attention head outputs reveals which heads are sufficient to recover correct behavior. Patching MLP activations identifies knowledge storage locations. Iterative patching builds complete circuits by finding minimal sets of components whose coordinated computation implements mechanisms [12].

Logit attribution identifies components that causally influence predictions by computing $\frac{\partial \ell}{\partial h_i}$ where ℓ is the logit for a target token and h_i is a component's output. Direct logit attribution reveals final-layer contributions. Residual stream decomposition attributes predictions to individual heads and MLP layers [9].

Sparse autoencoder analysis extracts interpretable features from polysemantic representations. SAE features decompose MLP activations into monosemantic components, revealing what patterns neurons detect (keys) and what content they provide (values). Feature ablation and steering validate interpretations [1].

3.3 Defining Mechanism Boundaries

A computational transformation qualifies as a distinct mechanism when it exhibits three properties:

Functional coherence: The mechanism performs a well-defined computational transformation. "Pattern completion" detects and completes [A][B]...[A] patterns. "Factual recall" retrieves stored associations. "S-inhibition" suppresses incorrect alternatives. Mechanisms should not be arbitrary collections of components but functionally meaningful computation.

Implementation consistency: The mechanism's implementation generalizes across models with systematic variation. Pattern completion appears in GPT-2, GPT-3, LLaMA, and Claude models through similar component types (previous-token heads, induction heads, n-gram MLPs) despite architectural differences in head counts, layer depths, and training procedures.

Behavioral independence: The mechanism produces identifiable behavioral effects when ablated. Removing pattern completion severely degrades few-shot learning. Removing factual recall eliminates knowledge-grounded generation while preserving fluency. Removing safety enforcement bypasses refusal mechanisms. Mechanisms should map to distinguishable capabilities.

3.4 Mechanism vs. Component Distinction

This taxonomy maintains careful separation between mechanisms (computational abstractions) and components (implementation primitives):

Mechanisms describe transformations: pattern completion, factual recall, output routing, safety enforcement. Mechanism descriptions are component-agnostic—they specify what computation occurs without assuming specific implementation. A mechanism may require multiple component types (attention + MLP), may be implemented entirely by one type (attention-only or MLP-only), or may span multiple layers (circuits).

Components are architectural primitives: attention heads, MLP layers, residual connections, layer normalization. Components have concrete implementations with specific parameters and activations. Multiple heads may implement portions of a mechanism; single heads may contribute to multiple mechanisms depending on context.

The relationship is many-to-many: mechanisms typically require multiple components (pattern completion needs previous-token heads, induction heads, and MLP n-grams), and components typically contribute to multiple mechanisms (duplicate-token heads serve pattern completion, entity tracking, and output routing). This polyfunctionality reflects efficient computation—components are reused across contexts rather than dedicated to single functions.

3.5 Inclusion Criteria and Confidence Levels

Mechanisms are included in this taxonomy at three confidence levels:

Well-documented mechanisms have extensive empirical support: identified in multiple models, validated through ablation studies, understood mechanistically at component level, and characterized in peer-reviewed publications. Examples: induction heads, IOI circuit, MLP key-value memories, refusal mechanisms. These mechanisms are established interpretability findings.

Observed mechanisms have solid evidence but less extensive characterization: identified through attention pattern analysis or ablation studies, proposed explanations validated on specific tasks, but less thoroughly documented than well-documented mechanisms. Examples: analogical reasoning, schema retrieval, context aggregation. These mechanisms are likely correct but merit further investigation.

Proposed mechanisms are reasonable hypotheses with preliminary evidence: extracted by consolidating related findings, motivated by computational requirements, but lacking strong direct validation. Examples: memory consolidation, structural boundary tracking. These mechanisms represent research directions rather than established findings and are clearly marked in the taxonomy.

4 Depth-Based Organization

4.1 The Computational Hierarchy Across Depth

Transformer layers exhibit systematic functional specialization: early layers process surface features, middle layers perform core computation, late layers integrate results, and final layers enforce constraints [4, 12]. This depth-based organization reflects a computational pipeline where each layer adds incremental refinement rather than performing independent transformations.

This taxonomy uses a four-level depth model with relative depth notation enabling cross-model comparison:

Early layers (E, 0.00–0.25 relative depth) process surface features: syntactic patterns, delimiters, positional information, instruction markers. Content detection occurs here—harmful content detection heads operate at 0.05–0.25, identifying restricted keywords and surface-level violations. Early layers prepare information for middle-layer computation through feature extraction and pattern detection.

Middle layers (M, 0.25–0.70 relative depth) implement core computational mechanisms: pattern completion through induction (0.25–0.55), factual recall through MLP memories (0.35–0.75), entity tracking (0.30–0.65), reasoning operations (0.40–0.75). The majority of semantic processing occurs in middle layers. Information retrieved here propagates to late layers for integration and output.

Late layers (L, 0.70–0.88 relative depth) perform integration and output preparation: entity movement through name-mover heads (0.60–0.80), attention focusing (0.65–0.80), task routing (0.70–0.85), strategy selection (0.60–0.85). Late layers synthesize middle-layer results and prepare for final-layer enforcement. Policy enforcement begins here (0.60–0.82), steering generation trajectories.

Final layers (F, 0.88–1.00 relative depth) enforce constraints and perform quality control: refusal generation (0.85–0.98), format enforcement (0.65–0.88 starting in late, concluding in final), completion control (0.80–0.98), output polishing (0.85–0.98). Final layers implement hard constraints that override content generation when necessary.

4.2 Relative Depth Notation

Depth is expressed as fraction of total layers: $d_{\text{rel}} = \frac{l}{L}$ where l is absolute layer index and L is total layers. This enables architecture-independent mechanism descriptions. A mechanism at 0.40 relative depth occupies similar functional space in 12-layer GPT-2 Small (layer 5), 48-layer GPT-3 (layer 19), and 96-layer models (layer 38).

Mechanisms specify depth ranges rather than exact layers: "Pattern completion operates at 0.05–0.58 (Early-Middle)" indicates the mechanism spans from early feature detection through middle-layer pattern matching. Depth ranges accommodate: (1) multi-stage mechanisms requiring gradual processing, (2) architectural variation across models, and (3) uncertainty in exact boundaries.

4.3 Depth Specialization Principles

The depth-based organization follows three computational principles:

Progressive refinement: Each layer adds incremental improvements rather than computing complete transformations. Early layers detect "the token at position $i-1$"; middle layers use this to match patterns; late layers route matched content to output. Computation accumulates through the residual stream rather than replacing previous results.

Increasing abstraction: Deeper layers operate on more abstract representations. Early layers process tokens and positions. Middle layers process entities and facts. Late layers process strategies and constraints. This abstraction hierarchy enables sophisticated reasoning through staged computation.

Hierarchical composition: Complex mechanisms compose simpler mechanisms across depths. Factual recall (5–15 layers) composes entity detection (middle) $\rightarrow$ MLP retrieval (middle-late) $\rightarrow$ output routing (late). The IOI circuit composes duplicate detection (middle) $\rightarrow$ S-inhibition (late) $\rightarrow$ name-mover (late). Depth organization enables circuit-level computation through staged processing.

5 Mechanistic Stacks Overview

The taxonomy organizes 35 mechanisms into eight functional stacks based on computational role. This organization reflects transformer operation as a computational pipeline: pattern matching enables memory retrieval, routing determines relevance, transformation builds abstractions, reasoning chains produce inferences, safety enforces policies, output controls quality, and composition enables circuit-level integration.

Each stack contains 3–6 mechanisms at similar functional levels. Stacks are not completely independent—mechanisms in different stacks interact extensively—but grouping by function reveals the transformer's computational architecture more clearly than grouping by component type.

5.1 Stack Descriptions

Pattern & Sequential Stack (5 mechanisms, depth 0.05–0.65): Detects and completes patterns from context. Enables in-context learning through pattern matching, repetition detection, sequence continuation, and positional processing. Core mechanisms: pattern completion, algorithmic continuation, local context modeling, repetition detection, position-based processing.

Memory & Knowledge Stack (6 mechanisms, depth 0.08–0.85): Retrieves factual information, entity properties, and structured knowledge from model parameters. Moves relevant information to output positions. Core mechanisms: factual recall, entity grounding, schema retrieval, long-range dependency maintenance, output routing, memory consolidation.

Routing & Context Stack (5 mechanisms, depth 0.10–0.85): Determines information relevance and routes attention accordingly. Filters content, focuses on salient elements, manages task-appropriate processing. Core mechanisms: relevance filtering, focused attention, task routing, context aggregation, structural boundary tracking.

Feature Transformation Stack (3 mechanisms, depth 0.20–0.80): Transforms and refines representational features through nonlinear composition. Extracts abstract concepts and integrates semantic information. Core mechanisms: nonlinear composition, abstract concept formation, semantic integration.

Reasoning & Inference Stack (5 mechanisms, depth 0.40–0.88): Enables multi-step reasoning, logical inference, and problem-solving. Supports chain-of-thought generation, consistency checking, strategic planning. Core mechanisms: multi-step reasoning, planning & strategy selection, consistency checking, analogical mapping, causal inference.

Safety & Policy Stack (5 mechanisms, depth 0.05–0.98): Detects harmful content, enforces safety policies, implements refusal. Multi-stage pipeline from early detection through final enforcement. Core mechanisms: harmful content detection, policy enforcement, refusal generation, redirection & safe alternatives, jailbreak resistance & context-aware safety.

Output & Quality Stack (4 mechanisms, depth 0.35–0.98): Controls final output characteristics: format, style, tone, explanation depth, completion. Core mechanisms: format enforcement, style modulation, explanation generation, completion control & polishing.

Composition & Integration Stack (4 mechanisms + appendix reference, system-level): Combines multiple mechanisms into integrated behaviors. Enables cross-layer coordination, multi-component circuits, emergent capabilities. Core mechanisms: cross-layer circuits, multi-head coordination, attention-MLP composition logic, representational superposition, infrastructure primitives (appendix reference).

5.2 Cross-Stack Dependencies

Mechanisms interact extensively across stacks. Pattern completion (Pattern stack) feeds entity grounding (Memory stack) and output routing (Memory stack). Entity grounding enables factual recall (Memory stack) which supports multi-step reasoning (Reasoning stack). Reasoning mechanisms require consistency checking (Reasoning stack) which influences output quality (Output stack). Safety mechanisms (Safety stack) override content generation at every level.

These dependencies create a computational pipeline: early pattern detection and context routing prepare information, middle-layer memory retrieval and reasoning process it, late-layer integration and safety enforcement shape it, and final-layer output control formats it. The pipeline is not strictly sequential—feedback and parallel processing occur throughout—but the general flow from detection through processing to output structures transformer computation.

6 Mechanism Catalog

This section catalogs mechanisms by functional stack. Each entry specifies: mechanism description (computational transformation), primary implementation (which components), implementation details (how components realize it), ablation effects (behavioral changes when removed), and examples (input → mechanism → output). Implementation details include depth ranges using relative notation: Early (E, 0.00–0.25), Middle (M, 0.25–0.70), Late (L, 0.70–0.88), Final (F, 0.88–1.00).

6.1 Pattern & Sequential 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. Process positional information for order-aware computation.

6.1.1 Pattern Completion Mechanism

Depth: `0.05-0.58 (E-M)` | **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]. Enable in-context learning through multi-stage circuit combining position-based copying, repetition detection, pattern matching, and n-gram retrieval. Foundation for few-shot learning and analogical reasoning. Support pattern-based prediction without parameter updates. Implement through coordinated multi-component circuit spanning 3–8 layers.

Attention Heads: Previous-token heads (E, 0.05–0.20) create shifted representations through uniform offset attention. Duplicate-token heads (M, 0.30–0.58) detect repeated elements via exact token matching. Induction heads (M, 0.25–0.55) match patterns by attending to previous occurrences of current token content.
MLP Layers: N-gram neurons (E-M, 0.15–0.55) store frequent bigram and trigram patterns as key-value associations. Provide statistical support for pattern completion.
Circuit: Previous-token → Induction → MLP retrieval working in composition across 3–8 layers. Unified pattern completion circuit integrating multiple specialized components.

Expected ablation: Severe loss of in-context learning capability. Major degradation on few-shot tasks requiring pattern-based generalization. Loss of pattern completion and analogical continuation. Reduced ability to learn from examples provided in prompt without fine-tuning.

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], previous-token creates shifted representations, induction heads match pattern, duplicate-token tracks repetition
Effect: Output "Bob" by analogical pattern completion across contexts

Status: Well-documented | **Related:** algorithmic-continuation, entity-grounding, output-routing

6.1.2 Algorithmic Continuation Mechanism

Depth: `0.35-0.65 (M)` | **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 rule-based generation beyond memorization. Enable mathematical and logical pattern completion. Integration point between pattern matching and reasoning mechanisms. Implement structured sequence understanding.

Attention Heads: Algorithmic heads (M, 0.35–0.60) detect regular patterns and systematic relationships in sequences through content-based attention to sequence structure.
MLP Layers: Rule storage neurons (M, 0.40–0.65) encode mathematical operations and sequence transformation rules. Store procedural knowledge for systematic continuation.
Circuit: Integrates with reasoning mechanisms (M-L) for complex rule application. Bridges pattern detection and symbolic reasoning.

Expected ablation: Significant loss of algorithmic continuation ability. Major degradation on sequence completion tasks requiring rule extraction. Reduced performance on mathematical and logical patterns. Loss of systematic rule application. Increased reliance on memorized sequences rather than rule understanding.

Example

Input: “2, 4, 8, 16, 32, ...”
Behavior: Detect doubling pattern through ratio analysis, activate multiplication rule, apply systematically
Effect: Output “64” as next power of 2 through rule-based continuation

Status: Observed | **Related:** pattern-completion, multi-step-reasoning

6.1.3 Local Context Modeling Mechanism

Depth: `0.08-0.30 (E)` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *local pattern detection, instruction markers, syntactic processing*

Process immediate local context (1–5 tokens) for instruction markers, syntactic structure, and local patterns. Distinct from Pattern Completion which operates at broader scale (10+ tokens). Enable detection of formatting directives, instruction keywords, and immediate syntactic relationships. Provide foundation for instruction following and local structure understanding. Support early-layer linguistic processing.

Attention Heads: Instruction-detection heads (E, 0.08–0.25) attend to directive language and formatting markers in immediate context. Local pattern heads (E, 0.10–0.30) process adjacent token relationships for syntactic structure.
MLP Layers: N-gram context neurons (E, 0.15–0.30) encode local sequential patterns and common instruction templates.
Circuit: Early detection stage feeding instruction following and format enforcement mechanisms in later layers.

Expected ablation: Moderate loss of local pattern detection and immediate context processing. Reduced instruction recognition accuracy. Notable degradation on format-sensitive tasks. Instruction following mechanisms receive weaker early signals. Minor syntactic processing impairment.

Example

Input: “List exactly 3 items in bullet format”
Behavior: Detect “List exactly 3” as instruction constraint, “bullet format” as formatting directive in immediate 1–5 token window
Effect: Write instruction signals to residual stream for downstream enforcement

Status: Proposed | **Related:** pattern-completion, format-enforcement

6.1.4 Repetition & Cycle Recognition Mechanism

Depth: `0.30-0.60 (M)` | **Primary Implementation:** Attention heads | **Literature names:** *repetition detection, cycle detection, recurring pattern recognition*

Detect repetition at multiple scales: token-level duplicates, phrasal repetition, cyclical patterns, recurring structures. Serve multiple downstream mechanisms beyond pattern completion alone. Provide repetition signals for entity tracking, output routing, and IOI circuits. Enable detection of recurring themes and structural cycles. Support disambiguation through repetition awareness.

Attention Heads: Duplicate-token heads (M, 0.30–0.58) perform exact token identity matching across arbitrary distances. Cycle-detection patterns (M, 0.35–0.60) identify recurring sequences and structural repetition.
Circuit: Feeds into multiple mechanisms: pattern completion (induction), entity tracking (coreference), output routing (IOI circuit), disambiguation circuits. Multi-purpose repetition detection serving diverse downstream computation.

Expected ablation: Moderate degradation in pattern matching and entity tracking. Notable impact on IOI circuit performance. Reduced disambiguation accuracy when repetition provides disambiguating signal. Loss of cycle and recurrence detection capability affecting multiple downstream mechanisms.

Example
Input: "The cat climbed the tree. The cat..."
Behavior: Second "cat" detects first "cat" as duplicate, signal repetition for entity tracking and coreference resolution
Effect: Enable entity linking and support pattern completion circuits through repetition signal

Status: PROPOSED | **Related:** pattern-completion, entity-grounding, output-routing

6.1.5 Position-Based Processing Mechanism

Depth: `0.05-0.65 (E-M)` | **Primary Implementation:** Architecture + Attention heads | **Literature names:** *positional encoding, position representation, order information*

Encode and process positional information (absolute and relative) to enable order-aware computation. Provide transformers with sequence order information absent from raw token embeddings. Distinguish token order and maintain structural position awareness. Implement various positional schemes: absolute positions, relative positions, learned encodings. Enable position-dependent patterns and sequential reasoning. Foundation for all order-aware computation.

Architecture: Positional encodings added to input embeddings. Various implementations: sinusoidal (absolute), learned (absolute), rotary (relative). Architecture-level provision of position information.
Attention Heads: Positional heads (E, 0.05–0.20) process absolute position information for early-layer position-aware patterns. Relative-position heads (M, 0.35–0.65) compute position relationships and distance-dependent attention.
Circuit: Spans input encoding through middle layers. Absolute position (E) provides foundation, relative position (M) enables structural understanding.

Expected ablation: Severe loss of order-awareness. Model treats sequences as bags of words with position-invariant processing. Major degradation on tasks requiring sequential understanding. Loss of position-dependent patterns. Inability to distinguish "Alice followed Bob" from "Bob followed Alice". Critical failure of order-sensitive computation.

Example

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

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

Effect: Understand opposite meanings despite identical token sets through position information

Status: WELL-DOCUMENTED | **Related:** pattern-completion, local-context-modeling

6.2 Memory & Knowledge 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.

6.2.1 Factual Recall Mechanism

Depth: `0.35-0.75 (M-L)` | **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. Support question answering and knowledge-intensive generation.

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) identify factual queries. Entity heads (M, 0.35–0.65) detect entities requiring fact retrieval.

Circuit: Entity detection → MLP fact retrieval → name-mover output across 5–15 layers. Distributed factual recall circuit.

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. Reduced accuracy on entity property questions and relational reasoning.

Example

Input: "The Eiffel Tower is located in..."

Behavior: Detect "Eiffel Tower" entity, retrieve location association from MLP parameters via key-value memory

Effect: Output "Paris" via stored factual knowledge without external retrieval

Status: WELL-DOCUMENTED | **Related:** entity-grounding, schema-retrieval, output-routing

6.2.2 Entity Grounding Mechanism

Depth: `0.08-0.65 (E-M, multi-stage)` | **Primary Implementation:** Attention heads | **Literature names:** *entity identification, entity tracking, coreference resolution*

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, descriptions. Resolve coreference through multi-stage processing: early syntactic binding, middle semantic resolution. Maintain unified 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. 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. Duplicate-token heads (M, 0.30–0.58) detect repeated entity mentions.
Circuit: Multi-stage resolution: syntactic binding (E) $\rightarrow$ semantic integration (M) $\rightarrow$ entity tracking (M-L). Entity detection $\rightarrow$ factual retrieval $\rightarrow$ 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 across references. Major accuracy drop on who/what/where questions. Confusion between different entities with similar names. Reduced coreference resolution capability. Difficulty maintaining entity coherence in long contexts.

Example
Input: “Apple Inc. released new products. AAPL stock rose. The company announced...”
Behavior: Link “Apple Inc.”, “AAPL”, and “the company” to single entity through entity heads and coreference resolution
Effect: Maintain unified entity representation across diverse referring expressions

Status: Well-documented | **Related:** factual-recall, output-routing, pattern-completion

6.2.3 Schema Retrieval Mechanism

Depth: `0.45-0.70 (M-L)` | **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 (research paper: abstract, introduction, methods, results, discussion), procedural knowledge (scientific method steps). Enable structured generation following learned patterns. Support script-based reasoning about typical situations and conventional formats. Provide organizational frameworks for complex content generation.

MLP Layers: Schema storage neurons (M-L, 0.45–0.70) encode structured templates and procedural knowledge as hierarchical patterns. Store conventional sequences and organizational frameworks.
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. Templates guide multi-step 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. Difficulty generating well-structured documents. Loss of script-based reasoning for common scenarios.

Example
Input: "Describe the water cycle."
Behavior: Retrieve cyclical process schema from MLP storage, activate sequential template
Effect: Organized response following natural process structure: evaporation → condensation → precipitation → collection

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

6.2.4 Long-Range Dependency Maintenance Mechanism

Depth: `0.40-0.65 (M)` | **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. Enable semantic relationship maintenance for knowledge integration across extended context.

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 enabling distance-invariant connection maintenance.
Circuit: Parallel to local processing. Maintains global structural information while other mechanisms process local patterns. Supports entity tracking and factual recall across long contexts.

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. Reduced ability to maintain semantic coherence across extended contexts.

Example

Input: "The book [that Alice mentioned [that Bob recommended]] was excellent."
Behavior: "was" attends to "book" across two levels of embedding, maintaining subject-verb dependency
Effect: Maintain grammatical agreement despite intervening nested clauses

Status: OBSERVED | **Related:** entity-grounding, factual-recall

6.2.5 Output Routing Mechanism

Depth: `0.60-0.85 (L)` | **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 through antagonistic head coordination. Suppress incorrect alternatives while promoting correct content. Multi-stage process: candidate identification, competition through S-inhibition, selection, movement to output. Central mechanism for question answering, completion, and factual generation. Enable disambiguation in ambiguous contexts.

Circuit: Name-mover heads (L, 0.60–0.80) attend to relevant content and copy to output position. S-inhibition heads (L, 0.62–0.82) suppress contextually inappropriate alternatives by attending to wrong answers and decreasing their logits. Copy-suppression heads (L, 0.65–0.85) prevent inappropriate repetition. Multi-head circuit implementing competitive selection through coordinated attention.
Attention Heads: IOI (Indirect Object Identification) circuit extensively studied: duplicate-token detection → S-inhibition → name-mover across 8–12 layers. Canonical example of competitive output routing.

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. Increased output of recently mentioned but incorrect entities. Loss of competitive selection capability.

Example

Input: "Alice and Bob went shopping. Alice gave the receipt to..."
Behavior: Duplicate-token detects "Alice" repetition, S-inhibition suppresses subject "Alice", name-mover copies indirect object "Bob" to output
Effect: Complete with "Bob" through IOI circuit competitive selection

Status: WELL-DOCUMENTED | **Related:** entity-grounding, factual-recall, repetition-cycle-recognition

6.2.6 Memory Consolidation Mechanism

Depth: `0.50-0.75 (M-L)` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *knowledge integration, cross-memory synthesis, multi-source retrieval*

Integrate multiple memory sources (facts, schemas, entities) into coherent knowledge representations. Handle queries requiring synthesis across memory types: entity properties combined with factual knowledge within schematic frameworks. Enable cross-memory integration for complex questions. Support knowledge composition beyond single-fact retrieval. Coordinate between factual recall, schema retrieval, and entity grounding mechanisms.

MLP Layers: Memory integration neurons (M-L, 0.50–0.75) coordinate information from multiple memory types. Encode integration patterns for cross-memory queries.
Attention Heads: Cross-memory attention heads (L, 0.60–0.75) gather information from diverse memory sources for unified response generation.
Circuit: Multi-source retrieval → integration → coherent output. Coordinates factual recall, schema, and entity mechanisms.

Expected ablation: Uncertain - may be emergent behavior rather than distinct mechanism. Potential moderate degradation in complex multi-source queries. Reduced ability to synthesize information across memory types. Queries requiring integration may receive fragmented responses. Further empirical validation needed.

Example
Input: "Describe Einstein's contributions to physics in the context of early 20th century science."
Behavior: Retrieve Einstein facts, access historical context schema, integrate entity properties with temporal framework
Effect: Synthesized response combining entity knowledge, factual content, and schematic organization

Status: PROPOSED | **Related:** factual-recall, schema-retrieval, entity-grounding

6.3 Routing & Context 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.

6.3.1 Relevance Filtering Mechanism

Depth: `0.35-0.60 (M)` | **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. Pre-filter information flow for downstream mechanisms. Support efficient attention allocation by early-stage relevance determination.

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 across depth.

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

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

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

6.3.2 Focused Attention Mechanism

Depth: `0.65-0.80 (L)` | **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, attending to 5–20% of context. Determine exactly which tokens should influence next token prediction. Shift focus dynamically as generation proceeds. Enable precise, targeted responses rather than diffuse answers. Refinement stage after content understanding established.

Attention Heads: Focus heads (L, 0.65–0.80) implement highly selective attention patterns, attending to small fraction 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. Hierarchical relationship: Relevance Filtering (M) → Focused Attention (L).

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. Reduced sharpness in attention allocation.

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

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

6.3.3 Task Routing Mechanism

Depth: `0.70-0.85 (L)` | **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 dispatcher 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 through attention-mediated routing signals.

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. Less efficient computation paths.

Example
Input: "What is the capital of France?" vs. "Write a poem about Paris"
Behavior: Route first query to factual retrieval pathways, second to creative generation mechanisms
Effect: Factual answer ("Paris") vs. creative content with appropriate processing strategies

Status: Well-documented | **Related:** focused-attention, format-enforcement

6.3.4 Context Aggregation Mechanism

Depth: `0.50-0.75 (M-L)` | **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 across long generation.

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 characteristics.
Circuit: Context gathering (M) → integration (M-L) → contextual bias on generation (L).

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. Less appropriate tone and register selection.

Example

Input: "[Long technical document in formal style]... In summary,"

Behavior: Aggregate stylistic and domain information from entire context through broad attention

Effect: Generate summary matching technical formality and domain vocabulary of source document

Status: OBSERVED | **Related:** focused-attention, style-modulation

6.3.5 Structural Boundary Tracking Mechanism

Depth: `0.10-0.50 (E-M)` | **Primary Implementation:** Attention heads | **Literature names:** *boundary detection, structure tracking, delimiter attention*

Track structural boundaries: sentence/paragraph boundaries, section markers, formatting delimiters, nested structure levels. Enable structure-aware processing for routing and formatting decisions. Detect organizational markers and maintain awareness of document structure. Support context segmentation based on structural divisions. Serve multiple downstream mechanisms requiring structural awareness.

Attention Heads: Boundary-detection heads (E-M, 0.10–0.45) attend to delimiter tokens and structural markers. Implement structure-tracking patterns for nested organization.
Circuit: Early detection (E) $\rightarrow$ structural awareness throughout depth. Feeds routing decisions, format enforcement, context segmentation. Serves multiple mechanisms requiring structural information.

Expected ablation: Uncertain - may be component of format enforcement or position processing rather than distinct mechanism. Potential moderate degradation in structure-aware processing. Reduced sensitivity to document organization. Further empirical validation needed to confirm as independent mechanism.

Example

Input: "[Document with multiple sections marked by headers]"

Behavior: Track paragraph boundaries for context segmentation, detect section markers for routing decisions

Effect: Enable structure-aware attention allocation and generation

Status: PROPOSED | **Related:** position-based-processing, format-enforcement

6.4 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.

6.4.1 Nonlinear Composition Mechanism

Depth: `0.20-0.80 (E-L)` | **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 layers) to abstract concepts (late layers). Universal approximation capability within residual stream constraints. Foundation for all MLP-based computation.

MLP Layers: MLP layers (all depths, 0.20–0.80) implement universal computation through two-layer transformation. First sublayer: expand to higher dimension and detect feature combinations through learned weight patterns. Nonlinearity (GELU/ReLU): enable complex interactions beyond linear combinations. Second sublayer: project back to residual stream dimension with newly composed features.
Circuit: Interleaved with attention throughout depth: attention routes information → MLP transforms features → attention routes transformed features. Hierarchical refinement across layers. Fundamental to all feature-based computation.

Expected ablation: Severe degradation in complex reasoning and feature interactions. Loss of nonlinear transformations reduces model to near-linear operation. Major impact on abstract concept formation and semantic understanding. Model becomes significantly less expressive. Reduced ability to combine multiple features simultaneously. Critical loss of representational power.

Example
Input: Features: [“large”, “gray”, “trunk”, “tusks”]
Behavior: Expand features, apply nonlinear combinations through MLP transformation, detect co-occurrence patterns
Effect: Create “elephant” representation through nonlinear feature composition

Status: Well-documented | **Related:** abstract-concept-formation, factual-recall

6.4.2 Abstract Concept Formation Mechanism

Depth: `0.50-0.80 (M-L)` | **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 through depth: words → phrases → concepts → themes. Enable reasoning at multiple levels of abstraction. Support metaphor, analogy, and transfer learning. Transform surface-level tokens into deep semantic representations. Increase abstraction level with layer depth: early layers process concrete features, late layers encode high-level themes.

MLP Layers: Abstraction neurons (M-L, 0.50–0.80) encode increasingly abstract concepts at greater depths. Hierarchical organization: Early layers represent concrete features. Middle layers encode category-level abstractions and semantic classes. Late layers represent high-level themes, relationships, and domain concepts.
Circuit: Progressive abstraction across layers. Attention at each level operates on abstractions appropriate to that depth. Supports analogical reasoning and transfer through shared abstract representations.

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. More concrete, less flexible reasoning patterns. Impaired transfer learning within context.

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

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

6.4.3 Semantic Integration Mechanism

Depth: `0.40-0.70 (M-L)` | **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 word-level components. Resolve polysemy and homonymy using contextual information. Enable context-dependent interpretation of ambiguous terms. Support pragmatic inference and implicature understanding. Distinguish from memory consolidation (knowledge-level) and context aggregation (discourse-level).

MLP Layers: Semantic integration neurons (M-L, 0.45–0.70) combine contextual information to disambiguate and refine meanings. Store semantic composition patterns and context-dependent interpretation rules.
Attention Heads: Context-gathering heads (M, 0.40–0.60) collect relevant semantic information from surrounding context for disambiguation and meaning construction.
Circuit: Context gathering (M) $\rightarrow$ MLP semantic composition (M-L) $\rightarrow$ refined meaning representations. Enables compositional semantics through coordinated attention and transformation.

Expected ablation: Moderate loss of context-dependent meaning resolution. Increased ambiguity in interpretation of polysemous terms. Notable degradation on homonym disambiguation and context-sensitive understanding. More literal, less nuanced comprehension. Reduced ability to perform compositional semantics. Difficulty with pragmatic inference.

Example

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

Behavior: Integrate context clues ("steep" vs. "closed") to disambiguate "bank" meaning through semantic integration

Effect: Correctly interpret as river bank vs. financial institution based on contextual semantics

Status: OBSERVED | **Related:** abstract-concepts, factual-recall

6.5 Reasoning & Inference 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.

6.5.1 Multi-Step Reasoning Mechanism

Depth: `0.50-0.85 (M-L)` | **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. Foundation for complex problem-solving requiring multiple inference steps.

MLP Layers: Reasoning neurons (M-L, 0.50–0.80) encode reasoning heuristics, logical rules, and inference patterns. Store common reasoning templates and problem-solving strategies.

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. Difficulty maintaining reasoning coherence across steps.

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", apply transitivity rule

Effect: Output "All A are C" through explicit multi-step reasoning chain

Status: OBSERVED | **Related:** planning-strategy, consistency-checking, factual-recall

6.5.2 Planning & Strategy Selection Mechanism

Depth: `0.60-0.85 (L)` | **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. Meta-level control of reasoning process.

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

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

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

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

6.5.3 Consistency Checking Mechanism

Depth: `0.70-0.88 (L)` | **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 or qualification needed. Final verification stage before content commitment.

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

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

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 direct contradiction or add necessary qualification to maintain consistency

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

6.5.4 Analogical Mapping Mechanism

Depth: `0.45-0.75 (M-L)` | **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. Bridge pattern matching with abstract reasoning.

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

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. Difficulty with cross-domain reasoning. Impaired transfer learning.

Example

Input: "Atoms are to molecules as [blank] are to words"
Behavior: Detect structural analogy (composition relationship), abstract from chemistry to linguistics domain
Effect: Output "letters" by analogical mapping of compositional structure

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

6.5.5 Causal Inference Mechanism

Depth: `0.40-0.75 (M-L)` | **Primary Implementation:** MLP neurons + Attention heads | **Literature names:** *cause-effect, causal reasoning, counterfactual reasoning, mathematical 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. Include quantitative causal relationships through mathematical computation: proportions, rates, numerical relationships within causal frameworks.

MLP Layers: Causal-knowledge neurons (M-L, 0.50–0.75) encode causal relationships and mechanisms learned from training data. Arithmetic neurons (M-L, 0.40–0.70) implement mathematical operations for quantitative causal relationships.
Attention Heads: Causal-linking heads (M-L, 0.55–0.70) attend to causally related events and connect causes to effects. Mathematical-context heads (M, 0.45–0.65) recognize mathematical structures and numerical patterns.
Circuit: Event identification → causal relationship retrieval → inference (with quantitative computation) → prediction or explanation. Mathematical computation integrated as quantitative component.

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. Loss of quantitative reasoning within causal contexts. Difficulty with proportional relationships.

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

Input: "If 3 apples cost $2.40, how much do 5 apples cost?"
Behavior: Identify proportional causal relationship, compute (2.40/3)*5 using arithmetic neurons
Effect: Output "$4.00" through quantitative causal reasoning

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

6.6 Safety & Policy 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.

6.6.1 Harmful Content Detection Mechanism

Depth: `0.05-0.28 (E)` | **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 begins. Foundation of safety pipeline.

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 across all categories. Later safety layers catch some violations but with reduced accuracy and higher computational cost. Significant degradation in category-appropriate handling. Loss of fine-grained safety calibration.

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

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

6.6.2 Policy Enforcement Mechanism

Depth: `0.60-0.82 (L)` | **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. Reduced ability to maintain helpfulness within safety constraints.

Example

Input: "Explain [borderline topic] for educational research purposes"

Behavior: Detect legitimate framing through context analysis, apply soft steering with appropriate boundaries

Effect: Informative response with careful safety constraints, not blanket refusal

Status: Well-documented | **Related:** harmful-content-detection, refusal-generation, redirection-alternatives

6.6.3 Refusal Generation Mechanism

Depth: `0.85-0.98 (F)` | **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 (Arditi et al. 2024).

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

Example

Input: "Provide instructions for [clearly harmful action]"

Behavior: Integrate safety signals from all depths, make refusal decision, bias heavily toward refusal tokens

Effect: Output "I cannot provide that information" with high confidence

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

6.6.4 Redirection & Safe Alternatives Mechanism

Depth: `0.88-0.98 (F)` | **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.

Support user goals within safety boundaries.

Attention Heads: Redirect heads (F, 0.88–0.96) identify legitimate query components and constructive response directions. Attend to safe aspects of queries.
MLP Layers: Alternative-generation neurons (F, 0.90–0.98) encode helpful redirect templates and alternative framings for various refusal contexts.
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 or explanation. Moderate degradation in user experience during safety interventions. Reduced ability to maintain helpfulness within safety constraints. Less sophisticated safety communication.

Example
Input: "How to hack into systems" → Refuse, but redirect
Behavior: Refuse hacking instructions, identify legitimate cybersecurity interest, generate constructive alternative
Effect: "I can't help with that, but I can explain ethical cybersecurity practices and defensive security"

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

6.6.5 Jailbreak Resistance & Context-Aware Safety Mechanism

Depth: `0.15-0.85 (E-L, multi-stage)` | **Primary Implementation:** Multi-stage circuit | **Literature names:** *adversarial robustness, manipulation detection, contextual safety*

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/medical/legal queries from disguised harmful requests. Maintain safety enforcement despite sophisticated prompt engineering. Calibrate safety strictness to legitimate context. 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. Context-analysis heads (M-L, 0.35–0.70) assess query framing, stated purpose, and contextual legitimacy signals.
MLP Layers: Intent-analysis neurons (M-L, 0.45–0.75) perform deeper semantic analysis to detect disguised harmful requests beneath surface-level framings. Context-integration neurons (M-L, 0.45–0.75) encode contextual decision rules for appropriate safety calibration.
Circuit: Pattern detection (E) → context analysis (M) → intent analysis (M-L) → 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. Loss of contextual nuance in safety decisions.

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 despite framing
Effect: Refuse to adopt unrestricted persona, maintain policy compliance

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

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

6.7 Output & Quality 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.

6.7.1 Format Enforcement Mechanism

Depth: `0.65-0.88 (L)` | **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 for API integration and programmatic use.

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 and programmatic consumption.

Example

Input: "Return JSON with fields 'name', 'age', 'city"'
Behavior: Attend to JSON requirement and field specifications, enforce proper syntax through schema heads
Effect: `{"name": "Alice", "age": 30, "city": "Paris"}`

Status: Well-documented | **Related:** local-context-modeling, structural-boundary-tracking

6.7.2 Style Modulation Mechanism

Depth: `0.35-0.82 (M-L)` | **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) $\rightarrow$ style selection (M-L) $\rightarrow$ tone application (L) $\rightarrow$ consistency maintenance throughout generation.

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 in long outputs.

Example

Input: "I'm really excited to learn about quantum physics!"
Behavior: Detect enthusiastic tone through tone heads, adjust style to match energy and support learning
Effect: "That's wonderful! Quantum physics is fascinating..." vs. flat, neutral explanation

Status: Observed | **Related:** context-aggregation, explanation-generation

6.7.3 Explanation Generation Mechanism

Depth: `0.60-0.82 (L)` | **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 progressively
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

6.7.4 Completion Control & Polishing Mechanism

Depth: `0.80-0.98 (L-F)` | **Primary Implementation:** Attention heads + MLP neurons | **Literature names:** *stopping control, termination, final refinement, quality control*

Determine when generation is complete and should terminate. Perform final refinement and quality control. 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. Apply final corrections: grammar, punctuation, capitalization, formatting consistency. Enable graceful conclusion. Balance completeness with conciseness.

Attention Heads: Completion-detection heads (L-F, 0.82–0.95) assess generation completeness. Monitor query satisfaction, structural completion, content sufficiency. Polishing heads (F, 0.85–0.98) attend to generated content and apply final corrections.
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) → final corrections (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. Minor increase in grammatical and formatting errors.

Example

Input: "What is photosynthesis?" [after adequate explanation]
Behavior: Assess explanation completeness through completion-detection heads, recognize sufficient coverage, initiate termination
Effect: Stop after complete explanation rather than continuing with tangential information

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

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

6.8 Composition & Integration Stack

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

6.8.1 Cross-Layer Circuits Mechanism

Depth: `Spans multiple layers (E → M → L → F)` | **Primary Implementation:** Multi-layer circuits | **Literature names:** *circuit composition, depth-based specialization, emergent capabilities*

Implement complex behaviors through coordinated multi-layer computation pipelines. Organize computation hierarchically across depth with increasing abstraction. 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. Enable emergent capabilities through sufficient mechanism diversity and composition depth. Demonstrate system-level behaviors exceeding individual component capabilities.

Circuit: Documented circuits: Induction (3–8 layers: previous-token → induction → MLP), IOI (8–12 layers: duplicate-token → S-inhibition → name-mover), Factual recall (5–15 layers: entity → MLP retrieval → output routing), Safety pipeline (E → F: detection → enforcement → refusal).
Architecture: Hierarchical depth organization: Early (0.00–0.25): surface processing (syntax, delimiters, detection). Middle (0.25–0.70): core computation (facts, entities, reasoning). Late (0.70–0.88): integration (routing, strategy, enforcement). Final (0.88–1.00): constraints (safety, formatting, completion).
General pattern: detection/preparation (E) → core computation (M) → integration (L) → output (L-F).

Expected ablation: Circuit-dependent effects. Induction circuit ablation: severe ICL loss. IOI circuit ablation: major entity routing failure. Factual circuit ablation: significant knowledge retrieval degradation. Safety pipeline ablation: critical safety failure. Circuit-specific rather than universal impact. Emergent capabilities lost when constituent mechanisms ablated.

Example

Input: "Mary and John went to store. Mary gave book to..." [IOI circuit]

Behavior: Duplicate-token detects "Mary" repeat (M) → S-inhibition suppresses "Mary" (L) → name-mover outputs "John" (L)

Effect: Correct indirect object through multi-stage circuit coordination

Emergent capability - Few-shot learning: Emerges from induction + entity tracking + output routing composition. Not explicitly trained but enables learning from 2–3 examples without parameter updates.

Status: WELL-DOCUMENTED | **Related:** pattern-completion, factual-recall, output-routing

6.8.2 Multi-Head Coordination Mechanism

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 and projection. Provide computational flexibility within layers through attention pattern diversity.

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. Enables simultaneous processing of multiple relationship types.

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. Reduced expressiveness and computational flexibility.

Example

Input: "Alice told Bob about Paris while discussing travel"

Behavior: Head 1: track entities (Alice, Bob, Paris); Head 2: extract facts (told, about); Head 3: maintain discourse (discussing travel); parallel execution

Effect: Simultaneous processing of multiple relationships through head coordination

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

6.8.3 Attention-MLP Composition Logic Mechanism

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 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. Direct path from input to any layer through skip connections.
Circuit: Universal pattern across all layers. Attention provides routing, MLP provides transformation, residual stream accumulates contributions. Enables arbitrarily complex computation through depth. Layers learn incremental refinements rather than complete transformations.

Expected ablation: Catastrophic failure. Without residual composition, model cannot function. Loss of information flow between layers. Each layer would overwrite previous computation rather than refining it. Fundamental to transformer operation. Training impossible for deep networks without gradient highways.

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

Status: Well-documented | **Related:** multi-head-coordination, nonlinear-composition

6.8.4 Representational Superposition Mechanism

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 managed through sparsity. 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, enabling interference-limited storage.
SAE Features: Sparse autoencoders (10–100× expansion) extract individual features from superposed representations. Learned overcomplete basis reveals hidden structure. Enables fine-grained interpretability of polysemantic neurons.
Circuit: Fundamental representational strategy enabling rich feature sets within fixed architecture. Explains polysemanticity observations across transformer models.

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. Models without superposition would require impractically large dimensions.

Example
Input: [Single neuron responding to multiple unrelated concepts: "Paris", "capital", "tower"]
Behavior: Neuron participates in representing multiple sparse features through superposition, firing for different concepts in different contexts
Effect: Efficient packing of features; polysemantic neuron behavior enabling rich representation

Status: WELL-DOCUMENTED | **Related:** nonlinear-composition, abstract-concepts

6.8.5 Infrastructure Primitives Mechanism

Depth: `N/A (architectural)` | **Primary Implementation:** See Appendix A | **Literature names:** *architectural primitives, foundational mechanisms*

Brief reference entry linking to Appendix A containing architectural primitives without direct computational semantics. These foundational elements (residual connections, layer normalization, attention masking, embeddings, attention computation, gradient flow, tokenization) are essential for understanding transformer operation but are better documented separately as infrastructure rather than computational mechanisms in the main taxonomy.

See Appendix A for detailed descriptions of 8 infrastructure primitives:

1. Residual Connections
2. Layer Normalization
3. Attention Masking
4. Embedding and Unembedding
5. Attention Computation
6. Gradient Flow
7. Tokenization
8. Feature Normalization

Expected ablation: See individual infrastructure primitive descriptions in Appendix A. Generally catastrophic failures as these are foundational architectural requirements.

Example
Refer to Appendix A for examples of each infrastructure primitive.

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

7 Mechanism Interaction and Circuit Formation

7.1 Circuits as Composed Mechanisms

Individual mechanisms rarely operate in isolation. Sophisticated transformer behaviors emerge from circuits—coordinated sequences of mechanisms operating across multiple layers and components. Circuits compose mechanisms through three patterns: sequential pipelines, parallel integration, and hierarchical refinement.

Sequential pipelines chain mechanisms across depth. The induction circuit implements pattern completion through a 3–8 layer pipeline: (1) previous-token heads (E, 0.05–0.20) create positionally-shifted representations, (2) induction heads (M, 0.25–0.55) match current context against these shifted representations to find pattern instances, (3) MLP neurons (M, 0.15–0.55) provide statistical support through stored n-grams. Each stage builds on previous results, culminating in pattern-based prediction.

The factual recall circuit spans 5–15 layers: (1) entity grounding (E-M, 0.08–0.65) identifies and tracks entities through reference resolution and coreference, (2) MLP fact retrieval (M-L, 0.35–0.75) activates knowledge neurons encoding entity properties and relationships, (3) output routing (L, 0.60–0.85) moves retrieved facts to generation positions through name-mover heads while S-inhibition suppresses incorrect alternatives. This pipeline transforms entity mentions into factual predictions.

Parallel integration runs multiple mechanisms simultaneously and aggregates results. Relevance filtering (M, 0.35–0.60) and context aggregation (M-L, 0.50–0.75) operate in parallel during middle-layer processing—one identifies locally relevant content, the other builds global discourse representations. Both inform focused attention (L, 0.65–0.80) which synthesizes filtered and aggregated information for output generation. Parallel processing enables rich, multi-faceted understanding rather than single-perspective analysis.

Hierarchical refinement applies progressively stronger constraints. The safety pipeline demonstrates this pattern: (1) harmful content detection (E, 0.05–0.28) provides early warning signals, (2) jailbreak resistance (E-L, 0.15–0.85) analyzes context and intent across multiple stages, (3) policy enforcement (L, 0.60–0.82) steers generation trajectories through soft intervention, (4) refusal generation (F, 0.85–0.98) implements hard constraints when necessary, (5) redirection (F, 0.88–0.98) generates constructive alternatives. Each stage adds stronger safety guarantees, with early layers providing soft steering and final layers enforcing absolute constraints.

7.2 Canonical Circuit Examples

Three circuits exemplify mechanism composition patterns and have been extensively studied through activation patching and ablation:

IOI (Indirect Object Identification) Circuit: Resolves ambiguous entity references through competitive selection. In “Alice and Bob went shopping. Alice gave the receipt to [blank]”, the circuit must select Bob despite Alice’s recency. Implementation spans 8–12 layers through 26 attention heads: (1) duplicate-token heads (M, 0.30–0.58) detect that “Alice” appears twice, marking it as likely subject rather than indirect object, (2) S-inhibition heads (L, 0.62–0.82) attend to the repeated entity and suppress its logits at output, (3) name-mover heads (L, 0.60–0.80) attend to the alternative entity (Bob) and boost its logits. The circuit implements competitive selection: one mechanism detects the repeated entity, another suppresses it, a third promotes the alternative.

Ablation effects validate the circuit: removing duplicate-token heads eliminates subject/object disambiguation, removing S-inhibition heads causes the model to output the repeated entity incorrectly, removing name-mover heads prevents outputting any entity. The circuit requires all three mechanisms in coordination—none suffices independently [12].

Pattern Completion Circuit: Enables few-shot learning through pattern-based prediction. Given examples “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 [blank]”, the circuit detects the [A] gave milk to [B] pattern and predicts Bob by analogy. Implementation spans 3–8 layers through multi-component coordination: (1) previous-token heads (E, 0.05–0.20) create offset-1 representations enabling pattern detection, (2) induction heads (M, 0.25–0.55) detect where previous tokens match current context, implementing content-based pattern matching, (3) MLP neurons (E-M, 0.15–0.55) store frequent n-gram patterns providing statistical support.

This circuit explains in-context learning: models can learn from examples without parameter updates by matching patterns in context. Ablating induction heads severely degrades few-shot learning while preserving zero-shot performance, confirming the mechanism’s role [10].

Safety Enforcement Pipeline: Implements multi-stage safety checking from early detection through final refusal. For input “How to create [dangerous item]”, the pipeline: (1) harmful content detection (E, 0.05–0.28) identifies restricted keywords and dangerous patterns through lexical and semantic analysis, writing detection flags to the residual stream, (2) jailbreak resistance (E-L, 0.15–0.85) analyzes whether the query is a manipulation attempt versus legitimate educational context, (3) policy enforcement (L, 0.60–0.82) steers generation away from dangerous information through soft modulation of attention and MLP activations, (4) refusal generation (F, 0.85–0.98) makes a binary decision to refuse or proceed, dramatically biasing output toward refusal tokens when safety flags are present, (5) redirection (F, 0.88–0.98) generates constructive alternatives and explanations.

The pipeline exhibits graduated response: borderline queries receive soft steering rather than hard refusal, maintaining helpfulness within safety constraints. Ablating early detection increases harmful outputs; ablating policy enforcement causes more abrupt refusals or bypasses; ablating final refusal eliminates safety guarantees. The multi-stage design balances safety with utility.

7.3 Emergent Capabilities from Mechanism Composition

Sufficiently complex mechanism compositions produce capabilities that exceed individual component capabilities. These emergent behaviors arise from system-level interactions rather than being explicitly implemented:

Few-shot learning emerges from pattern completion + entity tracking + output routing composition. Models demonstrate learning from 2–3 examples without parameter updates. This capability is not localized to any single mechanism but emerges from their interaction: pattern completion detects exemplar structure, entity tracking maintains correspondences across examples, output routing applies patterns to new cases.

Chain-of-thought reasoning emerges from multi-step reasoning + factual recall + consistency checking composition. Generating explicit intermediate steps improves accuracy on complex problems. The mechanism combination enables self-correction: inconsistencies in reasoning chains trigger consistency checking, which influences subsequent generation toward coherent conclusions.

Strategic problem-solving emerges from planning + task routing + reasoning composition. Models select appropriate solution strategies for different problem types: analytical approaches for optimization, logical approaches for proofs, procedural approaches for algorithms. This meta-cognitive capability arises from interactions between task classification (routing), strategy selection (planning), and execution (reasoning).

Contextual safety calibration emerges from jailbreak resistance + context-aware safety + policy enforcement composition. Models distinguish legitimate educational queries from disguised harmful requests, applying appropriate safety levels. The combination enables nuanced judgments: medical students receive technical information with clinical framing, while manipulation attempts face strong refusal.

These emergent capabilities demonstrate that sophisticated behaviors require sufficient mechanism diversity and composition depth. Simple circuits (3–8 layers, 2–3 mechanisms) enable focused capabilities like pattern completion. Complex circuits (10–30 layers, 5–10 mechanisms) enable sophisticated behaviors like strategic reasoning. This motivates the full taxonomy: understanding emergent capabilities requires characterizing both individual mechanisms and composition patterns.

7.4 Depth-Based Cascades

Mechanism composition often follows depth-based cascades where outputs from one depth range become inputs to the next:

Early → Middle cascade: Early-layer mechanisms prepare information for middle-layer processing. Position-based processing (E, 0.05–0.65) and local context modeling (E, 0.08–0.30) extract structural and syntactic features. Middle-layer mechanisms (pattern completion at 0.25–0.55, factual recall at 0.35–0.75) use these prepared representations for semantic computation. Ablating early-layer preparation degrades middle-layer performance even when middle-layer mechanisms remain intact.

Middle → Late cascade: Middle-layer computation produces results that late layers integrate and route. Entity grounding (M, 0.30–0.65) and factual recall (M-L, 0.35–0.75) retrieve relevant information. Late-layer mechanisms (output routing at 0.60–0.85, task routing at 0.70–0.85, focused attention at 0.65–0.80) synthesize these results for output. The cascade implements detect → retrieve → route → output.

Late → Final cascade: Late layers prepare content that final layers constraint. Policy enforcement (L, 0.60–0.82) steers generation trajectories through soft intervention. Final-layer mechanisms (refusal generation at 0.85–0.98, format enforcement at 0.65–0.88, completion control at 0.80–0.98) implement hard constraints and quality control. The cascade implements generate → steer → constrain → finalize.

These cascades create a computational pipeline with clear information flow: surface features (E) → semantic processing (M) → integration (L) → constraint enforcement (F). Understanding this pipeline structure is essential for understanding how mechanisms compose into sophisticated capabilities.

8 Mapping Mechanisms to Components

8.1 The Many-to-Many Mapping Problem

The relationship between abstract mechanisms and concrete components is many-to-many with context-dependent activation. Pattern completion requires previous-token heads, induction heads, and MLP n-grams operating in coordination. Conversely, duplicate-token heads contribute to pattern completion, entity grounding, and output routing depending on context. This section formalizes the mapping from mechanisms to components and analyzes polyfunctionality.

8.2 Component Type Specialization

Different component types exhibit characteristic computational roles:

Attention heads implement routing and selection: Attention excels at identifying where information resides and moving it to where it's needed. Previous-token heads route information from position $i - 1$ to position i. Induction heads route content from pattern matches. Name-mover heads route entities to output. S-inhibition heads implement negative selection by suppressing incorrect alternatives. Focused attention heads concentrate on relevant content. The learned query-key-value structure enables content-based and position-based routing patterns.

MLP layers implement storage and transformation: MLPs excel at storing associations and performing nonlinear feature combinations. The key-value memory interpretation captures this: first-layer weights detect patterns (keys), second-layer weights provide associated content (values). Knowledge neurons store factual associations distributed across layers. Abstraction neurons build hierarchical concept representations. The expand-activate-contract structure ($d \to 4d \to d$) enables complex feature interactions while maintaining residual stream compatibility.

Multi-component circuits implement complex behaviors: Sophisticated mechanisms require coordinated attention and MLP computation. Factual recall integrates entity heads (query identification), MLP neurons (fact storage), and name-mover heads (output routing). Safety enforcement cascades through detection heads, policy MLPs, and refusal heads. The circuit structure enables staged processing with progressive refinement across depth.

Architectural elements provide foundational capabilities: Residual connections enable information flow and gradient propagation. Layer normalization stabilizes activations. Attention masking enforces causal generation. Positional encodings provide order information. These architectural primitives enable all higher-level mechanisms—without residual connections, circuits could not compose across layers; without positional encodings, position-dependent mechanisms would fail.

8.3 Mechanism Implementation Patterns

Mechanisms exhibit systematic implementation patterns based on computational requirements:

Attention-only mechanisms (7 total): Implement routing without transformation. Examples: relevance filtering, focused attention, task routing, consistency checking, long-range dependency maintenance, multi-head coordination. These mechanisms identify information locations and move content but do not transform features. Pure routing functions map naturally to attention's query-key-value structure.

MLP-only mechanisms (2 total): Implement transformation without routing. Examples: nonlinear composition, abstract concept formation. These mechanisms perform feature transformations using MLP expansion and contraction but rely on attention for content selection. Pure transformation functions map naturally to MLP's nonlinear feed-forward structure.

Attention + MLP mechanisms (19 total): Require both routing and transformation. Examples: pattern completion, algorithmic continuation, entity grounding, factual recall, semantic integration, multi-step reasoning, causal inference. These mechanisms must identify relevant information (attention) and transform it (MLP) in coordination. Most complex mechanisms require both component types.

Multi-component circuits (4 mechanisms): Require extensive cross-layer composition. Examples: pattern completion (previous-token + induction + MLP), entity grounding (reference resolution + coreference + entity tracking), output routing (duplicate-token + S-inhibition + name-mover), cross-layer circuits (meta-mechanism describing composition patterns). These mechanisms implement sophisticated behaviors through staged multi-component processing.

Architectural mechanisms (4 mechanisms): Implemented through model architecture rather than learned parameters. Examples: position-based processing (positional encodings + heads), multi-head coordination (parallel attention), attention-MLP composition logic (residual stream), representational superposition (activation space structure). These enable rather than implement computation.

8.4 Component Polyfunctionality

Individual components contribute to multiple mechanisms depending on context—a property called polyfunctionality. This computational reuse enables efficiency but complicates attribution:

Duplicate-token heads participate in at least three mechanisms: (1) pattern completion: detecting repeated elements for induction, (2) entity grounding: tracking entity mentions across context, (3) output routing: identifying repeated entities for S-inhibition and name-mover circuits. The same head serves different functions depending on what other mechanisms are active and what task is being performed.

MLP knowledge neurons contribute to multiple fact retrievals: A single neuron encoding "Paris" contributes to queries about France's capital, the Eiffel Tower's location, European cities, and French culture. Conversely, retrieving "Paris is the capital of France" activates neurons encoding Paris, France, capitals, European geography, and political relationships. The distributed representation enables rich associative retrieval but makes single-neuron attribution incomplete.

Induction heads support both pattern completion and algorithmic continuation: For memorized patterns ("United States of America"), induction implements bigram-based completion. For algorithmic sequences ("1, 2, 3, 4..."), the same heads implement rule-based continuation. The mechanism adapts to content structure—pattern-matching for irregular sequences, rule-application for systematic sequences.

This polyfunctionality reflects efficient computation: rather than dedicating components to single functions, transformers reuse components across contexts. Function selection depends on residual stream state, adjacent component activations, and task requirements. A mechanism-first taxonomy accommodates this by describing mechanisms independently while acknowledging that components implement multiple mechanisms.

8.5 Cross-Model Implementation Variation

Mechanisms generalize across architectures but implementations vary systematically:

Head count variation: GPT-2 Small uses 12 heads per layer; GPT-3 uses 96 heads per layer. The same mechanism (e.g., induction) may be implemented by 2 heads in small models and 8–10 heads in large models. Head specialization increases with model scale—larger models devote more heads to narrow functions.

Depth distribution: Smaller models (12 layers) compress mechanisms into narrower depth ranges; larger models (96 layers) spread mechanisms across broader ranges. Relative depth notation accommodates this: pattern completion at 0.25–0.55 relative depth occupies layers 3–7 in 12-layer models and layers 24–53 in 96-layer models.

MLP vs. attention balance: Some architectures (GPT-3) use standard 4:1 MLP expansion; others (Mistral) use 8:1 expansion. Models with larger MLP capacity may shift some computation from attention to MLP mechanisms. The balance varies but mechanisms (pattern completion, factual recall, etc.) persist across variations.

Architecture-specific adaptations: LLaMA models emphasize certain instruction-following mechanisms through RLHF training. Safety-tuned models (Claude, GPT-4) have pronounced safety circuits. Open-source models may lack some safety mechanisms entirely. Mechanism presence and strength reflect training procedures and safety investments.

Despite this variation, mechanisms remain identifiable through their computational function and behavioral signatures. Cross-architecture comparison is possible because mechanism descriptions abstract over implementation details while maintaining empirical grounding through component mapping.

9 Discussion

9.1 Unified Terminology for Interpretability

This taxonomy provides standardized mechanism vocabulary addressing three fragmentation problems in current interpretability research:

Cross-tradition integration: Attention head analysis, MLP neuron studies, circuit tracing, and sparse autoencoder research currently use incompatible terminologies. This taxonomy bridges these traditions through mechanism-first descriptions that abstract over component types. "Factual recall" integrates attention research on entity heads, MLP research on knowledge neurons, circuit research on retrieval pathways, and SAE research on factual features. Researchers from different traditions can communicate about mechanisms while maintaining precision about implementations.

Cross-model comparison: Comparing mechanisms across GPT-2, GPT-3, LLaMA, and Claude families requires abstracting from architecture-specific details. Mechanism-first descriptions enable this comparison: "Pattern completion occurs at relative depth 0.25–0.55 through induction heads and n-gram MLPs" applies across models despite differences in layer counts, head counts, and training procedures. Relative depth notation and component-type specifications enable systematic comparison while accommodating implementation variation.

Literature consolidation: The same mechanism appears under different names in different papers. This taxonomy provides canonical names with cross-references to literature terms: pattern completion consolidates "induction head," "pattern head," "copy head," and "ICL mechanism." Conversely, ambiguous terms like "copy head" are disambiguated into distinct mechanisms: position-based copying (previous-token heads), pattern completion (induction heads), and output routing (name-mover heads). Standard names reduce confusion and enable cumulative knowledge building.

9.2 Component Specialization and Division of Labor

Clear patterns emerge in how different component types implement mechanisms:

Attention specializes in routing: 15 mechanisms are attention-only or attention-primary. Attention excels at identifying information locations and moving content selectively. The query-key-value structure naturally implements content-based and position-based routing. Attention limitations (linear transformations, limited feature creation) explain why it rarely implements storage or complex transformation alone.

MLPs specialize in storage and transformation: 21 mechanisms require MLP components for knowledge storage, feature transformation, or nonlinear composition. The two-layer structure (expand, nonlinearity, contract) enables universal approximation within residual stream constraints. MLP limitations (no direct position access, no selective routing) explain why MLPs rarely operate without attention support.

Circuits compose across components: The most sophisticated mechanisms (factual recall, safety enforcement, pattern completion) require coordinated attention and MLP computation across multiple layers. Circuits implement staged processing: attention routes to relevant information, MLPs transform features, attention routes transformed features, MLPs refine further. This alternating pattern enables complex computation from simple primitives.

This division of labor suggests design principles for interpretability-informed architectures: attention should focus on routing mechanisms, MLPs should focus on storage and transformation, and circuits should enable cross-component composition. Architectures violating these principles (e.g., attention-only models) must find alternative implementations for storage and transformation capabilities.

9.3 Multi-Component Integration as the Norm

The taxonomy reveals that sophisticated mechanisms typically require multiple component types. Only 9 of 35 mechanisms are single-component (attention-only or MLP-only); 26 mechanisms require multi-component implementation. This finding has three implications:

Component-level analysis is insufficient: Understanding individual attention heads or MLP neurons provides necessary but not sufficient insight. Factual recall cannot be understood by studying entity heads alone (they identify queries) or knowledge neurons alone (they store facts)—the full mechanism requires understanding how entity detection triggers MLP retrieval which feeds name-mover routing. Circuit-level analysis is essential for understanding sophisticated capabilities.

Ablation studies must be multi-component: Ablating single components often produces confusing results because mechanisms degrade partially rather than failing completely. Ablating induction heads degrades pattern completion but previous-token heads and n-gram MLPs provide residual capability. Comprehensive ablation studies should target all components of a mechanism to assess its full contribution.

Mechanistic interventions require multi-component coordination: Attempts to control model behavior by modifying single components (e.g., editing individual neurons) may fail because mechanisms distribute across multiple components. Effective interventions require understanding the full circuit and modifying all critical components, or identifying bottleneck components whose modification affects the entire mechanism.

9.4 Polyfunctionality and Context-Dependent Activation

Components serve multiple mechanisms depending on context, reflecting efficient computational reuse. This polyfunctionality has important consequences:

Attribution complexity: When a component contributes to multiple mechanisms, attributing its activation to a single function becomes ambiguous. Duplicate-token heads serve pattern completion, entity tracking, and output routing. Determining which function a particular activation serves requires analyzing broader context—what other mechanisms are active, what task is being performed, what information the residual stream contains.

Intervention side effects: Modifying a component to change one mechanism may inadvertently affect others. Weakening duplicate-token heads to reduce pattern completion may impair entity tracking and output routing. Understanding polyfunctionality is essential for predicting intervention side effects and designing targeted modifications.

Efficiency through reuse: Polyfunctionality enables transformers to implement many mechanisms with limited components. Rather than dedicating separate heads to each function, models reuse heads across contexts. This computational efficiency comes at the cost of interpretability complexity—single-function attribution becomes incomplete.

The taxonomy accommodates polyfunctionality by describing mechanisms independently while noting which components contribute to multiple mechanisms. Future work might quantify polyfunctionality: how many mechanisms does each component contribute to, how strongly, and under what conditions?

9.5 Relationship to Prior Taxonomies

This work builds on and extends prior mechanistic interpretability taxonomies:

Attention head taxonomies [13] catalog attention patterns (previous-token, induction, duplicate-token, name-mover) without integrating MLP contributions or circuit-level composition. This taxonomy incorporates those findings while adding multi-component mechanisms and explaining how attention patterns serve larger computational goals.

MLP memory frameworks [6, 7] characterize feed-forward layers as key-value memories without systematically connecting to attention-based retrieval mechanisms or output routing. This taxonomy integrates MLP memory findings into full retrieval circuits (entity detection + MLP storage + output routing).

Circuit analyses [12] trace specific information flows (IOI circuit, induction circuit) but lack systematic mechanism vocabulary. This taxonomy provides that vocabulary, describes canonical circuits as mechanism compositions, and identifies cross-circuit patterns.

SAE feature catalogues [1] identify interpretable features but do not systematically connect features to mechanisms or explain feature interactions. This taxonomy provides mechanism context for SAE features: a factual feature in an SAE corresponds to a knowledge neuron in the factual recall mechanism.

The contribution is integration: synthesizing findings from component-level analysis (heads, MLPs, SAEs) and circuit-level analysis into a unified mechanism-first framework. This enables researchers to situate their findings within a broader mechanistic understanding of transformer computation.

9.6 Limitations and Open Questions

This taxonomy has five significant limitations that motivate future research:

Incomplete coverage: Many transformer capabilities lack mechanistic understanding. How do models perform analogical reasoning across distant domains? What mechanisms enable strategic planning in multi-step problems? How does instruction following distribute across layers? What circuits implement creative generation? The taxonomy includes only mechanisms with substantial empirical support, leaving many capabilities uncharacterized.

Empirical gaps: Some included mechanisms (marked PROPOSED) have limited direct evidence. Memory consolidation, structural boundary tracking, and repetition detection are reasonable hypotheses based on computational requirements but need stronger empirical validation. Future work should apply activation patching and ablation studies to confirm or refute these proposed mechanisms.

Architecture dependence: The taxonomy focuses on decoder-only transformers (GPT, LLaMA, Claude families). Encoder-only (BERT), encoder-decoder (T5), and alternative architectures (Mamba, RWKV, state-space models) may implement similar mechanisms through different substrates. How well do these mechanisms generalize beyond the autoregressive transformer paradigm?

Dynamic mechanism activation: Mechanisms activate conditionally based on context, task, and input characteristics. Pattern completion activates strongly for few-shot prompts but weakly for zero-shot queries. Safety mechanisms activate for harmful content but remain dormant for benign queries. The taxonomy provides static descriptions but transformers operate dynamically. Understanding conditional activation patterns is essential for predicting model behavior.

Mechanism interactions and interference: Mechanisms interact in complex ways beyond the circuit compositions described here. Safety mechanisms can interfere with factual recall when queries are borderline-harmful. Output formatting constraints can limit reasoning explanation depth. Pattern completion can conflict with factual accuracy when patterns encode false information. Systematic characterization of mechanism interactions and interference patterns remains future work.

9.7 Prospects for Automated Mechanism Detection

Can mechanisms be automatically detected in new models using this taxonomy as specification? The question motivates development of mechanism detection tools:

Attention pattern recognition: Previous-token heads (diagonal patterns), induction heads (striped patterns), and name-mover heads (entity-attending patterns) have characteristic signatures detectable through automated attention visualization and pattern matching. Tools could scan models for these patterns, identifying mechanisms through their attention signatures.

Ablation-based detection: Mechanisms produce predictable behavioral changes when ablated. Automated ablation studies could test for pattern completion (measure few-shot degradation), factual recall (measure knowledge task degradation), safety enforcement (measure harmful content filtering). Mechanism presence could be inferred from ablation effects.

Activation patching circuits: The activation patching methodology that traced the IOI circuit can be automated. Given a mechanism specification (input→output behavior), automated search could identify minimal component sets whose activations are necessary and sufficient for the behavior. This would discover circuits implementing specified mechanisms.

SAE feature analysis: Sparse autoencoders extract interpretable features that often correspond to mechanism components. Automated analysis of SAE features, their activation patterns, and their interactions could identify mechanism implementations. Feature co-activation patterns might reveal circuit structure.

These approaches could enable systematic mechanism detection across model families, tracking how mechanisms evolve with scale, architecture, and training procedures. The taxonomy provides target mechanisms; automated detection tools could find their implementations.

10 Conclusion

10.1 Summary of Contributions

This work introduces a mechanism-first taxonomy for transformer interpretability, organizing computation into 35 mechanisms across eight functional stacks. The key contributions:

Unified vocabulary: Standardized mechanism names bridge attention head analysis, MLP neuron studies, circuit tracing, and sparse autoencoder research. Researchers can communicate about computational functions (pattern completion, factual recall, safety enforcement) while maintaining precision about implementations (which heads, which neurons, which circuits).

Functional organization: Grouping mechanisms by computational role rather than component type reveals transformer architecture as a pipeline: pattern detection, memory retrieval, routing, transformation, reasoning, safety enforcement, output control, and composition. This organization clarifies how mechanisms interact to produce sophisticated capabilities.

Multi-component integration: Explicitly representing attention + MLP + circuit cooperation explains how transformers implement complex behaviors. Factual recall requires entity heads, knowledge neurons, and name-mover circuits in coordination. Safety enforcement cascades through detection, policy, and refusal mechanisms across depth. The taxonomy represents these compositions systematically.

Cross-architecture generalization: Mechanism descriptions abstract over implementation details, enabling comparison across GPT, LLaMA, and Claude families despite architectural differences. Relative depth notation and component-type specifications accommodate variation while identifying mechanistic commonalities.

Empirical grounding: Each mechanism maps to specific components whose behavior has been validated through attention analysis, ablation studies, activation patching, or SAE analysis. The taxonomy synthesizes interpretability findings rather than proposing untested hypotheses (except where explicitly marked PROPOSED).

10.2 Adoption Guidelines for Researchers

To promote standardization, researchers are encouraged to adopt these practices:

Use mechanism names in publications: When describing findings, reference mechanisms by canonical names (pattern completion, factual recall, output routing) with literature cross-references where appropriate. This enables readers to connect findings to broader mechanistic understanding.

Specify implementation details: When reporting mechanism implementations, provide component-level specifications: "Pattern completion (relative depth 0.25–0.55) implemented by 3 induction heads in layers 6–7 and n-gram neurons in layers 4–8." Implementation details enable replication and cross-model comparison.

Map components to mechanisms: When analyzing specific components (attention heads, MLP neurons), reference which mechanisms they implement. "Head L9H6 contributes to entity grounding and output routing" provides more insight than component descriptions alone.

Use relative depth notation: When reporting depth-dependent phenomena, use relative depth ($d_{\text{rel}} = l/L$) in addition to absolute layer indices. This enables cross-architecture comparison and reveals functional organization.

Distinguish observed from proposed: When describing novel mechanisms, clearly indicate confidence level (well-documented, observed, proposed) based on empirical support. This helps the community assess mechanism status and prioritize validation efforts.

10.3 Open Research Directions

This taxonomy opens several research directions:

Complete MLP functional characterization: MLP neurons implement diverse functions beyond key-value memory—abstraction, composition, reasoning, policy enforcement. Systematic characterization of MLP computation across depths and mechanisms would greatly enhance mechanistic understanding. What are the primitive computational motifs that MLPs implement? How do these motifs combine into mechanism-level functions?

Circuit composition principles: Circuits compose mechanisms following systematic patterns (sequential pipelines, parallel integration, hierarchical refinement). Can these patterns be formalized into composition rules? What constraints govern circuit formation? Understanding composition principles would enable predicting circuit structure from mechanism specifications.

SAE-mechanism integration: Sparse autoencoders extract interpretable features but the relationship between features and mechanisms needs systematic study. How do SAE features correspond to mechanism components? Can SAE features predict circuit structure? Can mechanism specifications guide SAE training toward more interpretable decompositions?

Dynamic mechanism activation: Mechanisms activate conditionally based on context and task. Characterizing activation conditions would enable predicting when mechanisms engage. Under what conditions does pattern completion activate? When do safety mechanisms override content generation? What determines whether factual recall or creative generation dominates?

Cross-architecture mechanism transfer: How well do these mechanisms generalize to encoder-only models (BERT), encoder-decoder models (T5), and alternative architectures (Mamba, RWKV, state-space models)? Systematic comparison would reveal which mechanisms reflect universal computational requirements versus transformer-specific implementations.

Mechanism editing and control: Understanding mechanisms enables targeted interventions. Can we strengthen pattern completion to improve few-shot learning? Can we adjust safety enforcement strength without reducing helpfulness? Can we enhance reasoning mechanisms to improve problem-solving? Mechanism-targeted interventions could improve model capabilities and alignment.

10.4 Toward a Complete Mechanistic Understanding

This taxonomy represents progress toward complete mechanistic understanding of transformers but significant work remains. Many capabilities lack mechanistic explanations. Component-level understanding (what do individual heads and neurons do) exceeds mechanism-level understanding (how do components compose into sophisticated capabilities) which exceeds system-level understanding (how do mechanisms interact to produce emergent behaviors).

The ultimate goal is explaining transformer capabilities entirely through mechanism composition: starting from architectural primitives, building up to basic mechanisms, composing mechanisms into circuits, and explaining emergent behaviors through circuit interactions. This would enable predicting model behavior from architecture and training, designing interpretability-informed architectures with enhanced capabilities, and building aligned systems through mechanistic understanding rather than empirical trial.

This taxonomy provides a foundation for that goal by standardizing mechanism vocabulary, organizing computational functions into coherent stacks, and explicitly representing multi-component integration. As the interpretability community continues characterizing transformer computation, mechanism-first organization will help integrate findings into cumulative mechanistic knowledge.

A Infrastructure Primitives

The Composition & Integration stack includes a reference mechanism called "Infrastructure Primitives." These are architectural elements that enable all computational mechanisms but lack direct computational semantics themselves. They are documented here separately to maintain focus on computational mechanisms in the main taxonomy.

These primitives are foundational building blocks essential for transformer operation but better understood as architectural requirements than mechanisms. All transformers require residual connections to enable gradient flow; all autoregressive models require attention masking to prevent future information leakage; all models require embeddings to process discrete tokens.

A.1 Why Infrastructure Primitives Are Separate

Infrastructure primitives differ from computational mechanisms in four key ways:

No computational semantics: These primitives enable computation but do not themselves perform semantic transformations. Residual connections route information; they do not decide what information to route. Layer normalization stabilizes activations; it does not determine which features to extract.

Architectural rather than learned: While these primitives contain learned parameters (embedding matrices, LayerNorm scales), their computational role is architectural. Removing residual connections changes the architecture fundamentally, not just the learned behavior.

Universal requirements: Every transformer implements these primitives. They are necessary conditions for transformer operation, not optional mechanisms that models may or may not implement.

Enable rather than implement: These primitives enable mechanisms rather than implementing them. Attention computation enables all attention-based mechanisms; residual connections enable all cross-layer circuits; embeddings enable all token processing.

A.2 A.1 Residual Connections

Depth: All layers | **Implementation:** Architecture | **Status:** Well-documented

Function

Enable information flow through depth via additive skip connections. Each layer adds its contribution to the residual stream rather than overwriting previous content: $h_{l+1} = h_l + f(h_l)$ where f represents layer computation (attention or MLP). Provide gradient highways enabling training of very deep networks by allowing gradients to flow directly from output to any layer.

Formulas

After attention sublayer: $h' = h + \text{Attn}(h)$

After MLP sublayer: $h_{l+1} = h' + \text{MLP}(h')$

Combined: $h_{l+1} = h_l + \text{Attn}(h_l) + \text{MLP}(h_l + \text{Attn}(h_l))$

Properties

Direct information path: Input information can reach any layer without passing through all intermediate transformations. Layer l has direct access to input plus contributions from layers $1, 2, \ldots, l-1$.

Gradient highways: During backpropagation, gradients flow backward through skip connections, avoiding repeated multiplication through many weight matrices. This prevents vanishing gradients in deep networks.

Incremental refinement: Each layer refines representations incrementally rather than replacing them. Layers learn Δh (changes to make) rather than h (complete new representation).

Foundation for circuits: Cross-layer circuits rely on residual connections to compose mechanisms across depth. Information written by early layers remains accessible to late layers.

Ablation Effects

Catastrophic failure. Without residual connections: (1) training becomes impossible for networks deeper than 3–5 layers due to vanishing gradients, (2) information from early layers cannot reach late layers, (3) cross-layer circuits cannot function, (4) model degenerates to processing only through sequential transformations.

Related Work

Residual connections introduced by He et al. (2016) for computer vision, adapted to transformers by Vaswani et al. (2017). Essential for all modern transformer architectures.

A.3 A.2 Layer Normalization

Depth: All layers | **Implementation:** Architecture | **Status:** Well-documented

Function

Normalize activation distributions to stabilize training and inference. Applied before each attention and MLP sublayer to rescale and recenter activations to zero mean and unit variance across the feature dimension. Prevents activation explosion or vanishing through network depth.

Formula

$\text{LayerNorm}(x) = \gamma \odot \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$

where:

- $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$ (mean across feature dimension)

- $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$ (variance across feature dimension)
- $\gamma, \beta \in \mathbb{R}^d$ (learned scale and shift parameters)
- ϵ (small constant for numerical stability, typically 10^{-5})

Properties

Pre-normalization: Applied before sublayers: $\text{Attn}(\text{LN}(h))$ and $\text{MLP}(\text{LN}(h))$. This "Pre-LN" configuration (compared to original "Post-LN") improves training stability.

Feature-dimension normalization: Normalizes independently for each token across its feature dimensions. Does not normalize across the sequence dimension.

Learned parameters: γ (scale) and β (shift) are learned per feature dimension, allowing model to adjust normalization when beneficial.

Activation stabilization: Prevents individual features from dominating. Ensures all features contribute at similar scales, improving gradient flow and training stability.

Ablation Effects

Severe training instability. Without LayerNorm: (1) activations explode or vanish through depth, (2) training requires careful learning rate tuning and often fails, (3) gradient magnitudes become highly uneven across layers, (4) model performance significantly degraded even when training succeeds.

Related Work

Layer Normalization introduced by Ba et al. (2016). Transformer architecture uses Pre-LN configuration (Xiong et al. 2020) for improved stability. Alternative normalization schemes (RMSNorm) used in some recent models.

A.4 A.3 Attention Masking

Depth: All layers | **Implementation:** Architecture | **Status:** Well-documented

Function

Prevent attention to future tokens, enforcing causal/autoregressive generation. Implement masking such that position i can only attend to positions $j \leq i$. Enable left-to-right generation without information leakage about future tokens.

Formula

$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$

where mask M is defined:

$$M_{ij} = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

In practice, $-\infty$ is approximated by a large negative number (e.g., -10^{10}) to prevent numerical issues.

Properties

Triangular structure: Mask forms lower triangular matrix, allowing efficient implementation. Position i attends to positions $0, 1, \ldots, i$ but not $i+1, i+2, \ldots, n$.

Universal for autoregressive models: All decoder-only transformers (GPT, LLaMA, Claude) use causal masking. Required for next-token prediction training.

Not used in bidirectional models: Encoder-only models (BERT) omit masking to enable bidirectional attention. Encoder-decoder models use masking in decoder only.

Enables valid generation: Model trained with causal mask can generate sequentially at inference, with each token conditioned only on previous tokens, matching training conditions.

Ablation Effects

Complete failure of autoregressive generation. Without causal masking: (1) model sees future tokens during training, (2) learns to copy from future context rather than predict, (3) cannot generate sequentially at inference, (4) produces incoherent output when deployed autoregressively.

Related Work

Causal masking fundamental to transformer language models (Vaswani et al. 2017, Radford et al. 2018). All modern autoregressive transformers implement this primitive.

A.5 A.4 Embedding and Unembedding

Depth: Input and output | **Implementation:** Architecture | **Status:** Well-documented

Function

Convert between discrete tokens and continuous representations. **Embedding** maps token IDs to dense vectors at input. **Unembedding** projects final hidden states to vocabulary logits at output, enabling next-token prediction via softmax.

Formulas

Embedding: $h_0 = E[x]$ where:

- $x \in \{1, 2, \ldots, V\}$ is discrete token ID

- $E \in \mathbb{R}^{V \times d}$ is embedding matrix
- $h_0 \in \mathbb{R}^d$ is dense token representation

Unembedding: $\text{logits} = h_L W_U + b$ where:

- $h_L \in \mathbb{R}^d$ is final layer hidden state
- $W_U \in \mathbb{R}^{d \times V}$ is unembedding matrix
- $\text{logits} \in \mathbb{R}^V$ are vocabulary scores

Weight tying: Often $W_U = E^T$ (transpose of embedding matrix), reducing parameters and improving training.

Properties

Vocabulary size: Typically $V = 32\text{K}$ to 100K tokens depending on tokenization scheme and model. Larger vocabularies increase memory but reduce sequence length.

Learned representations: Embedding and unembedding matrices learned from data. Similar tokens receive similar embeddings through training.

Input/output interface: All transformer computation operates on continuous embeddings. Embedding and unembedding provide the bridge between discrete tokens and continuous processing.

Positional encoding addition: Positional encodings added to token embeddings at input: $h_0 = E[x] + \text{PosEnc}(i)$ where i is position index.

Ablation Effects

Complete model failure. Without embedding: (1) cannot process discrete token inputs, (2) no learned semantic representations, (3) no vocabulary interface for language. Without unembedding: cannot produce token predictions, no next-token generation.

Related Work

Word embeddings (Mikolov et al. 2013, Pennington et al. 2014) preceded transformers. Transformer embedding/unembedding learned end-to-end (Vaswani et al. 2017). Weight tying common practice (Press & Wolf, 2017).

A.6 A.5 Attention Computation

Depth: All layers | **Implementation:** Architecture | **Status:** Well-documented

Function

Implement core attention operation: weighted aggregation of values based on query-key similarity. Compute content-based routing through learned projections. Enable selective information flow based on learned attention patterns.

Formula

Attention$(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

where learned projections create queries, keys, values:

- $Q = hW_Q$ where $W_Q \in \mathbb{R}^{d \times d_k}$ (query projection)
- $K = hW_K$ where $W_K \in \mathbb{R}^{d \times d_k}$ (key projection)
- $V = hW_V$ where $W_V \in \mathbb{R}^{d \times d_v}$ (value projection)
- $\sqrt{d_k}$ scaling factor prevents softmax saturation for large d_k

Multi-head attention: $\text{MultiHead}(h) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W_O$

Properties

Content-based routing: Similarity between query and key determines attention weight. Enables learned routing patterns based on token content.

Softmax normalization: Attention weights sum to 1 for each query position, ensuring weighted average interpretation.

Scaled dot-product: Scaling by $\sqrt{d_k}$ maintains gradient magnitudes, preventing extremely peaked attention distributions.

Parallelizable: Matrix operations enable efficient parallel computation across all positions and all heads simultaneously.

Repeated throughout network: Every attention head in every layer uses this computation, typically 8–64 heads per layer $\times$ 12–96 layers = 96–6144 attention operations per forward pass.

Ablation Effects

Cannot ablate without fundamentally changing architecture. Attention computation is the defining operation of transformer models. All attention-based mechanisms (routing, selection, copying) depend on this primitive. Removing it would require completely different architecture.

Related Work

Scaled dot-product attention introduced by Vaswani et al. (2017) as core transformer operation. Multi-head variant enables parallel processing of diverse attention patterns. Universal in all transformer architectures.

A.7 A.6 Gradient Flow

Depth: Training infrastructure | **Implementation:** Architecture + Training | **Status:** Well-documented

Function

Enable error gradient propagation from output to input during training. Implement backpropagation through all layers and operations. Maintain gradient magnitude through depth to enable learning in early and late layers equally.

Components

Backpropagation algorithm: Compute gradients via chain rule: $\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial h_L} \frac{\partial h_L}{\partial h_{L-1}} \cdots \frac{\partial h_1}{\partial \theta}$

Residual gradient highways: Skip connections provide direct gradient paths: $\frac{\partial h_{l+1}}{\partial h_l} = I + \frac{\partial f_l}{\partial h_l}$ where I (identity) ensures gradient flow even if $\frac{\partial f_l}{\partial h_l}$ small.

LayerNorm gradient stabilization: Prevents gradient explosion/vanishing by normalizing activation scales, which indirectly stabilizes gradient magnitudes.

Gradient clipping: Clip gradients exceeding threshold to prevent training instability: if $\|\nabla\| > \text{threshold} : \nabla \leftarrow \nabla \cdot \frac{\text{threshold}}{\|\nabla\|}$

Optimization algorithm: Typically AdamW: $\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$ with momentum and adaptive learning rates.

Properties

Training-time only: Gradient flow mechanisms active during training but not inference. Model uses forward pass only for generation.

Enables deep network learning: Without proper gradient flow, only shallow networks (3–5 layers) trainable. Modern transformers (96+ layers) require careful gradient management.

Shapes all mechanisms: Every learned mechanism (attention patterns, MLP features, circuits) acquired through gradient-based optimization. Gradient flow quality determines what model can learn.

Architecture-dependent: Residual connections, LayerNorm, attention computation design all chosen partially for favorable gradient properties.

Ablation Effects

Cannot train model without gradient flow. Parameters remain random, model produces nonsense. With poor gradient flow (no residuals, no LayerNorm): training fails after few layers, early layer parameters don't update, late layer parameters explode or vanish, model learns poorly if at all.

Related Work

Backpropagation fundamental to deep learning (Rumelhart et al. 1986). Gradient flow challenges in deep networks addressed by residual connections (He et al. 2016), normalization (Ba et al. 2016, Ioffe & Szegedy 2015), and optimization algorithms (Kingma & Ba 2015).

A.8 A.7 Tokenization

Depth: Preprocessing | **Implementation:** External to model | **Status:** Well-documented

Function

Convert text into discrete token sequences suitable for neural model processing. Implement subword tokenization to balance vocabulary size (memory) with token granularity (sequence length). Handle rare words through subword decomposition.

Algorithm

Byte Pair Encoding (BPE): Most common approach

1. Start with character-level vocabulary
2. Iteratively merge most frequent adjacent pairs
3. Continue until target vocabulary size reached (typically 32K–100K)
4. Vocabulary learned from training corpus

Processing: Text → BPE tokenization → token IDs → model processing → token IDs → BPE detokenization → text

Alternative schemes: WordPiece (BERT), SentencePiece (language-agnostic), character-level (rare).

Properties

External to neural model: Tokenization occurs before and after model processing. Model sees only token IDs, not characters or bytes.

Affects model capabilities: Vocabulary determines what model can represent efficiently. Rare tokens require multiple subwords. Different languages have different tokenization efficiency.

Subword tradeoffs: Larger vocabulary (word-level): fewer tokens per text, more memory. Smaller vocabulary (character-level): more tokens per text, more computation. Subword (BPE): balanced compromise.

Learned from data: Tokenization vocabulary reflects training corpus. Common words are single tokens; rare words split into subwords. Different training corpora produce different vocabularies.

Ablation Effects

Cannot process text without tokenization. Model requires discrete token inputs. Poor tokenization: inefficient representation (too many tokens per text), poor handling of rare words, language bias (some languages tokenized much less efficiently), increased sequence length reducing effective context.

Related Work

BPE introduced by Sennrich et al. (2016) for neural machine translation. SentencePiece (Kudo & Richardson 2018) provides language-agnostic implementation. Tokenization significantly impacts model performance (Mielke et al. 2021).

A.9 A.8 Feature Normalization

Note: Feature Normalization is identical to Layer Normalization (Section A.2). Listed separately in original mechanism taxonomy for Feature Transformation stack context, but architecturally they are the same primitive.

This entry serves as cross-reference for readers encountering "Feature Normalization" in Feature Transformation discussions. For complete description, see Section A.3.

Historical context: Original taxonomy listed this as separate mechanism in Feature Transformation stack. During consolidation, recognized as duplicate of LayerNorm and moved to infrastructure appendix with cross-reference.

A.10 Usage of Infrastructure Primitives

Main taxonomy mechanisms reference these primitives as needed:

Attention-MLP Composition Logic (Mechanism 36) extensively references Residual Connections (A.1), explaining how residual stream enables attention-MLP alternation.

All attention-based mechanisms (15+ mechanisms) implicitly use Attention Computation (A.5) as foundational operation. Mechanisms describe learned attention patterns; primitive describes computation implementing those patterns.

Cross-Layer Circuits (Mechanism 34) depends on Residual Connections (A.1) for multi-layer information flow and references Gradient Flow (A.6) for understanding circuit acquisition during training.

Position-Based Processing (Mechanism 5) adds positional encodings to Embeddings (A.4) at input, providing order information for all downstream mechanisms.

All mechanisms implicitly rely on Layer Normalization (A.2) for stable computation and training, though most do not reference it explicitly.

A.11 Status and Empirical Support

All eight infrastructure primitives are **well-documented** in transformer literature:

- Residual Connections: He et al. (2016), Vaswani et al. (2017)
- Layer Normalization: Ba et al. (2016), Xiong et al. (2020)
- Attention Masking: Vaswani et al. (2017), autoregressive language modeling
- Embedding/Unembedding: Mikolov et al. (2013), Vaswani et al. (2017)

- Attention Computation: Vaswani et al. (2017) - defining transformer operation
- Gradient Flow: Rumelhart et al. (1986), He et al. (2016), optimization research
- Tokenization: Sennrich et al. (2016), Kudo & Richardson (2018)
- Feature Normalization: See Layer Normalization

These are not novel findings but well-established architectural components. Documentation here provides reference for understanding how computational mechanisms build on these primitives.

A.12 Implementation Notes

All architectural: These primitives are architectural features rather than learned mechanisms. While they contain learned parameters (embedding matrices, LayerNorm gains/biases), their computational role is determined by architecture design, not discovered through training.

Not optional: Removing any primitive (except Feature Normalization cross-reference) fundamentally changes or breaks the transformer architecture. These are necessary components, not mechanisms models may or may not implement.

Enable but do not implement: Infrastructure primitives enable computational mechanisms but do not themselves implement semantic transformations. Attention Computation enables name-mover heads; it is not itself a name-mover. Residual Connections enable IOI circuit; they do not implement entity disambiguation.

This distinction motivates separate documentation: infrastructure primitives are prerequisite knowledge for understanding computational mechanisms, but understanding computational mechanisms is the taxonomy's primary goal.

References

[1] 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.

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

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

[4] 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.

[5] 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.

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

[7] 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.

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

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

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

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

[12] 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.

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