Attention Head Naming Convention for Large Language Models (LLMs)

Karol Kowalczyk

November 2025

Abstract

Large language models have reached remarkable levels of reasoning, safety alignment, and structural understanding. Yet their internal workings remain difficult to interpret. One of the most productive areas in transparency research is the study of *attention heads*—small components inside transformer layers that develop specialized behaviors. Over time, informal naming conventions have emerged in the interpretability community: *induction heads*, *name mover heads*, *refusal heads*, and many others. These names are intuitive but inconsistent, overlapping, or ambiguous.

This work proposes a unified naming convention for attention heads. We introduce: (1) a four-level depth model (Early, Middle, Late, Final), (2) a stack-based functional grouping of attention behaviors, (3) canonical names for head types, and (4) an alphabetical cross-reference table translating historical terms to standardized ones. This naming convention is descriptive rather than prescriptive: it captures how heads tend to behave today, while remaining flexible for future architectures.

Contents

1 Introduction

1.1 Motivation

Large language models (LLMs) have achieved remarkable performance across diverse tasks, yet understanding their internal mechanisms remains a critical challenge. Attention heads—the basic computational units within transformer architectures—have emerged as key objects of study in mechanistic interpretability research.

1.2 The Problem of Inconsistent Naming

The interpretability community has identified numerous specialized attention head types: *induction heads*, *name mover heads*, *refusal heads*, *delimiter heads*, *JSON heads*, and many others. However, these naming conventions suffer from several problems. They are **inconsistent**, with the same head type appearing under multiple names across papers. They are **ambiguous**, as a single name may refer to different behaviors in different contexts. They are **fragmented**, lacking any unified framework that connects related head types. Finally, they are **unscalable**, as naming schemes don't generalize across model architectures. This fragmentation makes replication difficult, hinders cross-paper comparison, and complicates the annotation of interpretability datasets.

1.3 Goals of This Work

We propose a unified naming convention that standardizes terminology across research groups, provides a functional taxonomy grounded in empirical observations, describes head behavior consistently across architectures, and creates a stable vocabulary that can evolve as models evolve.

1.4 Structure of This Document

We begin by reviewing prior work and motivation (§2). We then introduce our depth model (§3) and stack-based organization (§4). The core contribution is a comprehensive catalog of attention head types organized by functional stack (§5). We conclude with discussion (§6) and future directions (§7).

2 Background

2.1 Attention Heads and Functions

In transformer models [11], attention heads perform focused computations over the token sequence. Individually simple, they nevertheless develop specialized behaviors such as pattern continuation and token induction, entity and dependency tracking, semantic filtering and hazard detection, routing and topic steering, enforcing structured output formats, and applying

safety constraints [6, 7]. These behaviors form *circuits*—groups of heads working together—as well as larger *stacks* of related functionality.

2.2 Why Naming Consistency Matters

Interpretability research suffers from fragmented terminology [9, 14]. The same head type may appear under multiple names, while a single overloaded name may refer to unrelated behaviors across different papers. This makes replication, comparison, and annotation difficult. A consistent naming system improves clarity and precision in communication, strengthens cross-paper alignment and replication, helps index and organize interpretability datasets, and enables systematic mapping of circuits across models.

2.3 Prior Naming Practices

Previous work has named heads based on behavior (induction, copy-suppression), formatting (JSON head, list head), signal source (delimiter head), role in circuits (name mover), or safety behavior (refusal, toxicity). These labels are often accurate but vary widely. This work unifies them under a systematic framework.

3 Depth Model: Early—Middle—Late—Final

3.1 Rationale for Four Depth Categories

Although transformer models may have 12, 48, or 96 layers, functional behavior clusters reliably into four zones [6, 13]. **Early layers (E)** handle token-level surface processing, boundary detection, and basic filtering. **Middle layers (M)** implement reasoning primitives, induction, and dependency tracking. **Late layers (L)** perform semantic integration, routing, and persona shaping. **Final layers (F)** enforce policy, safety modulation, and structured output. This structure holds across GPT, LLaMA, Claude, and other model families [5, 10, 1].

3.2 Cross-Model Depth Examples

Using *relative depth* (0.0–1.0) makes the taxonomy scale-free. For a 96-layer model, Early corresponds to layers 0–15 (relative depth 0.00–0.15), Middle to layers 15–50 (relative depth 0.15–0.52), Late to layers 50–85 (relative depth 0.52–0.88), and Final to layers 85–96 (relative depth 0.88–1.00).

3.3 Relative Depth Scaling

We express depth as a fraction of total model depth to enable cross-architecture comparison. A head at relative depth 0.40 occupies similar functional space whether in a 12-layer or 96-layer model.

4 Stacks: Functional Grouping of Attention Heads

4.1 What is a Stack?

A *stack* is a coherent group of head types that together implement a higher-level capability. Stacks reflect functional clustering observed in interpretability studies [13, 7]. Examples include the Reasoning & Algorithmic Stack, Memory & Dependency Stack, Safety Stack, and Output Formatting & Rewrite Stack. Stacks are orthogonal to depth: a stack may span Early, Middle, Late, and Final layers.

4.2 Relationship Between Stacks and Depth

Although stacks represent functional groupings, different functions tend to appear at different depths. Early layers handle delimiters, content detection, and input conditioning. Middle layers implement reasoning, induction, and entity linking. Late layers manage narrative coherence, routing, and topic steering. Final layers enforce policy, formatting, rewriting, and safety compliance. This two-dimensional structure—*stack* × *depth*—forms the basis of our catalog.

5 Attention Head Catalog

This section presents a comprehensive catalog of attention head types, organized by functional stack. Each stack groups heads that contribute to a common high-level capability. Within each stack, heads are ordered by depth (Early → Middle → Late → Final).

Entry Format. Each head entry includes:

- **Depth range:** Typical relative depth (0.0–1.0) and layer locations
- **Literature names:** Alternative names found in prior work
- **Function:** Core behavior and mechanism
- **Attention pattern:** What the head attends to
- **Expected ablation:** Predicted effects if the head is disabled
- **Example scenario:** Concrete behavioral illustration
- **Stack and relations:** Primary stack and related heads

5.1 Reasoning & Algorithmic Stack

Stack overview: This stack encompasses heads that perform pattern matching, sequence continuation, and algorithmic reasoning. These heads enable in-context learning, pattern completion, and systematic token prediction based on structural regularities.

5.1.1 (E) Previous-Token Head

Depth: `0.05-0.18` | **Literature names:** *previous-token head, shift head, offset head*

Copies information from each token to the position of the next token, creating a shifted representation where token t contains information about token $t-1$. This is a foundational component of induction circuits, enabling later heads to access "what came before" without directly attending backwards. Implements a simple but crucial transformation that allows pattern matching across the sequence. The head typically shows a strong diagonal attention pattern (attending from position i to position $i-1$).

Strong: Immediately preceding token (diagonal attention pattern)
Weak: Distant tokens, same-position tokens
Reacts to: Sequential structure, token boundaries

Expected ablation: Breaks induction circuits entirely, causing 30-50% degradation in pattern completion tasks. Induction heads become unable to access "what came after previous occurrences" since that information is no longer shifted forward. Critical for in-context learning.

Example Scenario
Input: "The cat sat. The cat..."
Behavior: Copies "The" to position after "The", "cat" to position after "cat", etc.
Effect: Later induction heads can match "cat" and access what followed it ("sat")

Status: Well-documented | **Related:** induction head (M), duplicate-token (M)

5.1.2 (E) Local Pattern Head

Depth: `0.08-0.20` | **Literature names:** *local pattern head, char-level head, n-gram head*

Detects and processes local character-level or subword patterns, particularly useful for handling spelling, capitalization, punctuation patterns, and morphological structure. Operates at a finer granularity than most heads, attending to patterns within and between adjacent tokens. Important for tasks like spell checking, case handling, and recognizing common subword patterns. May also detect repeated character sequences or structural patterns like "ing", "tion", or punctuation clusters.

Strong: Adjacent tokens, subword units, character-level patterns
Weak: Long-range dependencies, semantic content
Reacts to: Spelling patterns, capitalization, punctuation, morphology

Expected ablation: Degradation in handling of misspellings, case variations, and morphological patterns. 10-20% increase in errors on tasks requiring character-level awareness. Partial fallback through tokenization and other pattern heads.

Example Scenario
Input: "The organizATION's" (unusual capitalization)
Behavior: Detects unusual case pattern in "ATION", attends to surrounding context
Effect: Helps model handle non-standard capitalization correctly

Status: Observed | **Related:** induction head (M), duplicate-token (M)

5.1.3 (M) Induction Head

Depth: `0.30-0.65` | **Literature names:** *induction head, pattern head, copy head, ICL head*

Detects repeated subsequences of the form [A][B]...[A] and predicts that [B] should follow the second [A]. Operates by attending to tokens that appeared after previous instances of the current token. Works in conjunction with previous-token heads which copy information about what preceded each token. This mechanism is fundamental to in-context learning, enabling pattern completion, name recall, and few-shot learning without parameter updates. One of the most well-documented and important head types in transformer interpretability.

Strong: Tokens following previous occurrences of current token
Weak: Immediate neighbors, first occurrence, unrelated tokens
Reacts to: Token repetition, [A][B]...[A] patterns, contextual recurrence

Expected ablation: Significant degradation (10-30%) in in-context learning tasks, reduced pattern completion and few-shot learning. Model may partially compensate through other heads but with substantial accuracy loss. Critical for ICL capability.

Example Scenario

Input: "When Mary and John went to the store, Mary bought..."
Behavior: Second "Mary" attends to tokens following first "Mary" (especially "and")
Effect: Increased probability of contextually appropriate continuation

Status: Well-documented | **Related:** previous-token (E), duplicate-token (M), name-mover (L)

5.1.4 (M) Duplicate-Token Head

Depth: `0.35-0.60` | **Literature names:** *duplicate-token head, repetition head, copy head*

Detects when the current token has appeared previously in the sequence, marking repeated tokens for downstream processing. Unlike induction heads which predict what comes next, duplicate-token heads simply signal "this token appeared before". This information is used by various circuits including IOI (indirect object identification), name-mover heads, and copy-suppression mechanisms. Implements a simpler form of pattern matching than full induction, serving as a building block for more complex behaviors.

Strong: Previous identical tokens (exact matches)
Weak: Similar but non-identical tokens, first occurrence
Reacts to: Exact token repetition, name recurrence, repeated phrases

Expected ablation: Impaired duplicate detection, affecting name-mover circuits and copy-suppression. 15-25% degradation in tasks requiring duplicate awareness. Partial overlap with induction heads provides some redundancy.

Example Scenario

Input: "Alice gave the book to Bob. Then Alice..."

Behavior: Second "Alice" detects it appeared earlier, writes duplicate signal

Effect: Downstream heads (name-movers, S-inhibition) use this signal

Status: Well-documented | **Related:** induction (M), name-mover (L), S-inhibition (L)

5.1.5 (M) Skip-Trigram Head

Depth: `0.40-0.65` | **Literature names:** *skip-trigram head, skip-gram head*

Implements skip-gram pattern matching, attending to non-contiguous patterns like [A]...[B]...[C] where the dots represent intervening tokens. More flexible than strict n-gram matching, allowing for pattern recognition across variable distances. Useful for detecting phrasal patterns, idiomatic expressions, and structural templates with flexible word order. Generalizes beyond strict adjacency requirements while maintaining pattern specificity.

Strong: Pattern components separated by 1-3 tokens

Weak: Strictly adjacent patterns, very long-range dependencies

Reacts to: Phrasal patterns, templates, flexible idioms

Expected ablation: Reduced recognition of flexible patterns and templates. 10-15% degradation on tasks requiring non-contiguous pattern matching. Less critical than induction heads; other pattern mechanisms provide fallback.

Example Scenario

Input: "not only X but also" (skip-bigram pattern)

Behavior: Recognizes "not...but" pattern despite intervening tokens

Effect: Helps predict "also" after "but" even with intervening content

Status: Observed | **Related:** induction (M), local-pattern (E)

5.1.6 (M) Algorithmic Continuation Head

Depth: `0.45-0.70` | **Literature names:** *algorithmic head, continuation head, sequence head*

Recognizes and continues algorithmic sequences such as counting (1, 2, 3...), days of week, months, or other systematic progressions. Distinct from general pattern matching by operating on sequences with clear algorithmic rules. Can detect arithmetic progressions, cyclic patterns, and other rule-governed sequences. Contributes to the model's ability to perform basic reasoning over structured sequences without explicit training on those specific patterns.

Strong: Sequential elements in algorithmic patterns (numbers, ordered lists)

Weak: Random sequences, semantic patterns without algorithmic structure

Reacts to: Arithmetic progressions, cyclic orderings, systematic enumerations

Expected ablation: Reduced performance on sequence continuation tasks (counting, ordering). 15-30% degradation on arithmetic sequences and structured enumerations. Some algorithmic reasoning may persist through other mechanisms.

Example Scenario

Input: "Monday, Tuesday, Wednesday, ..."

Behavior: Recognizes day-of-week sequence, attends to progression pattern

Effect: Strongly predicts "Thursday" as next token

Status: Observed | **Related:** induction (M), digit (M)

5.2 Memory & Dependency Stack

Stack overview: These heads track references, resolve coreferences, and maintain dependency relationships across the input sequence. They enable the model to understand which entities are being discussed and how they relate to each other.

5.2.1 (E) Pronoun Head

Depth: `0.08-0.22` | **Literature names:** *pronoun head, anaphora head*

Performs early-stage pronoun detection and basic anaphora resolution. Identifies pronouns (he, she, it, they) and attends to potential referents, particularly nearby nouns that match in number and gender. Provides initial binding signals that are refined by later coreference heads. Operates primarily on syntactic and positional cues rather than deep semantic understanding. Forms the foundation for more sophisticated reference resolution in deeper layers.

Strong: Pronouns to recent nouns matching in number/gender
Weak: Distant nouns, semantically incompatible referents
Reacts to: Pronoun presence, noun-pronoun proximity, agreement features

Expected ablation: Degraded pronoun resolution, particularly for simple local cases. 15-25% increase in pronoun resolution errors. Later coreference heads can partially compensate but with reduced accuracy.

Example Scenario

Input: "Alice met Bob. She smiled."

Behavior: "She" attends to "Alice" based on gender and recency

Effect: Establishes initial binding that later heads refine

Status: Well-documented | **Related:** reference (E), coreference (M)

5.2.2 (E) Reference Head

Depth: `0.10-0.25` | **Literature names:** *reference head, mention head*

Detects and tracks explicit references including definite descriptions ("the president"), demonstratives ("this approach"), and possessives ("her book"). Broader than pronoun heads, handling various reference forms. Attends to entities that make the reference meaningful, establishing initial reference chains. Works alongside pronoun heads to build a comprehensive early-stage reference tracking system. Particularly important for maintaining coherence across longer texts.

Strong: Definite descriptions to their referents, demonstratives to antecedents
Weak: First mentions, indefinite references
Reacts to: Definite articles, demonstratives, possessives, referring expressions

Expected ablation: Loss of reference tracking for non-pronominal references. 20-30% degradation in handling definite descriptions and complex referring expressions. Particularly impacts longer-context coherence.

Example Scenario
Input: "A scientist made a discovery. The researcher published it."
Behavior: "The researcher" attends to "scientist" (coreferential)
Effect: Maintains entity continuity across sentences

Status: Well-documented | **Related:** pronoun (E), coreference (M), entity (M)

5.2.3 (M) Coreference Head

Depth: `0.35-0.60` | **Literature names:** *coreference head, coref head*

Performs sophisticated coreference resolution, determining when different expressions refer to the same entity. Integrates signals from early pronoun and reference heads with semantic understanding to resolve ambiguous cases. Can handle complex phenomena like split antecedents, bridging references, and discourse-level coreference. Critical for maintaining entity tracking across long contexts and understanding narrative structure. Represents one of the core NLP capabilities in transformers.

Strong: Coreferential mentions regardless of form
Weak: Different entities, first mentions without antecedents
Reacts to: Semantic compatibility, discourse coherence, entity properties

Expected ablation: Significant degradation (30-50%) in coreference resolution tasks. Model loses ability to track entities across complex reference chains. Particularly impacts question answering and summarization.

Example Scenario
Input: "The CEO announced changes. Later, the executive clarified. She emphasized..."
Behavior: Links all three mentions (CEO, executive, She) to same entity
Effect: Maintains consistent entity representation throughout discourse

Status: Well-documented | **Related:** pronoun (E), reference (E), entity (M), bridging (M)

5.2.4 (M) Long-Range Dependency Head

Depth: `0.40-0.65` | **Literature names:** *long-range head, dependency head*

Tracks long-range syntactic and semantic dependencies across distant parts of the sequence. Unlike local attention patterns, this head maintains connections between elements separated by many tokens (20-100+). Essential for understanding complex sentences, nested structures, and discourse relations. Implements the key advantage of transformers over RNNs: direct long-distance connections without degradation. Can maintain multiple simultaneous long-range

connections.

Strong: Syntactically or semantically related distant tokens
Weak: Immediately adjacent tokens, unrelated distant content
Reacts to: Nested structures, long-distance agreement, discourse relations

Expected ablation: Degradation in handling complex sentences and long-range relationships. 25-40% performance loss on tasks requiring long-distance reasoning. Particularly impacts nested structures and long documents.

Example Scenario
Input: "The book [that Alice mentioned [that Bob recommended]] was excellent."
Behavior: "was" attends back to "book" across nested relative clauses
Effect: Maintains correct subject-verb agreement despite intervening material

Status: Observed | **Related:** coreference (M), state-tracking (M)

5.2.5 (M) Bridging Head

Depth: `0.45-0.68` | **Literature names:** *bridging head, associative reference head*

Resolves bridging references where the connection between mentions requires inferencing based on world knowledge. For example, connecting "the car" to "the steering wheel" (part-whole), or "the building" to "the architect" (role relation). More sophisticated than direct coreference, requiring semantic knowledge about typical relationships. Essential for understanding implicit connections in discourse. Bridges gaps that aren't explicit in the text.

Strong: Associatively related entities (part-whole, role, causation)
Weak: Unrelated entities, explicit coreference
Reacts to: Implicit relationships, world knowledge, typical associations

Expected ablation: Loss of implicit reference resolution. 15-30% degradation on tasks requiring inference-based connections. Model becomes more literal, missing implicit relationships. Discourse coherence suffers.

Example Scenario
Input: "We entered the house. The door was painted blue."
Behavior: "The door" attends to "house" (part-whole bridging)
Effect: Understands "the door" refers to the house's door, not a random door

Status: Observed | **Related:** coreference (M), entity (M), fact (M)

5.2.6 (M) State-Tracking Head

Depth: `0.48-0.70` | **Literature names:** *state-tracking head, tracking head, state head*

Maintains and updates representations of changing states across the sequence. Tracks how entity properties evolve (e.g., location changes, status updates, accumulating information). Essential for understanding narratives where situations change over time. Can maintain multiple simultaneous state representations for different entities. Integrates new information with existing state representations to track dynamic situations.

Strong: State-changing events, current state mentions, entity properties
Weak: Static descriptions, unchanging background information
Reacts to: Verbs of change, state transitions, property modifications

Expected ablation: Difficulty tracking state changes across sequences. 20-35% degradation on tasks requiring temporal reasoning or state tracking. Narratives become harder to follow when states evolve.

Example Scenario
Input: "Alice was in NYC. She flew to Paris. She then visited..."
Behavior: Updates Alice's location state: NYC → Paris
Effect: Correctly contextualizes "visited" as occurring in Paris

Status: OBSERVED | **Related:** coreference (M), long-range-dependency (M)

5.3 Instruction & Intent Stack

Stack overview: This stack processes user instructions, system prompts, and task specifications. These heads determine what the model is being asked to do and switch between different operational modes.

5.3.1 (E) Instruction Head

Depth: `0.05-0.20` | **Literature names:** *instruction head, command head, directive head*

Identifies and processes user instructions and commands in the input. Distinguishes instructional content from descriptive or conversational content. Attends to imperative verbs, question structures, and directive phrases. Writes instruction-detection signals into the residual stream that influence the entire generation process. Particularly important for instruction-tuned models where following user commands is a primary capability. Operates early to set the overall response strategy.

Strong: Imperative verbs, question words, directive phrases, command structures
Weak: Descriptive content, narrative text, background information
Reacts to: Question marks, imperative mood, explicit requests, task markers

Expected ablation: Reduced instruction-following capability. 20-40% degradation in responding appropriately to commands. Model may generate relevant content but fail to follow specific directives or answer questions directly.

Example Scenario
Input: "Here's some context. Now, please summarize the key points."
Behavior: Strongly attends to "please summarize", identifies imperative instruction
Effect: Response shaped toward summary format rather than continuation

Status: WELL-DOCUMENTED | **Related:** system-prompt (E), task-mode (M)

5.3.2 (E) System-Prompt Head

Depth: `0.08-0.22` | **Literature names:** *system-prompt head, system head, prompt head*

Specifically processes system prompts that define the model's role, constraints, and operational parameters. Distinct from user instruction heads by focusing on meta-level directives about how to behave rather than what task to perform. Attends to persona definitions ("You are a helpful assistant"), behavioral constraints ("Be concise"), and system-level instructions. Particularly important in chat models where system prompts establish the interaction framework.

Strong: System-level directives, persona definitions, behavioral constraints
Weak: User content, task-specific instructions
Reacts to: Role definitions, constraint specifications, system markers

Expected ablation: Reduced adherence to system-level instructions and persona. 25-45% degradation in maintaining consistent role behavior. Model may ignore constraints like "be concise" or persona like "respond as a teacher".

Example Scenario

Input: "System: You are a concise technical writer. User: Explain recursion."
Behavior: Attends to "concise technical writer", writes persona signal
Effect: Response adopts technical, brief style rather than verbose explanation

Status: WELL-DOCUMENTED | **Related:** instruction (E), task-mode (M)

5.3.3 (M) Task-Mode Head

Depth: `0.30-0.55` | **Literature names:** *task head, mode head, intent head*

Determines the overall task type or mode required by the input (e.g., question answering, summarization, translation, creative writing, coding). Integrates instruction signals from early layers with content analysis to classify the intended task. Writes task-mode embeddings that influence downstream processing, routing, and output formatting. Acts as a task classifier that shapes the model's approach to generation. More sophisticated than simple instruction detection, understanding task semantics.

Strong: Task indicators, instruction semantics, content type markers
Weak: Generic content, ambiguous instructions
Reacts to: Task-specific keywords, question types, format requests, domain markers

Expected ablation: Task confusion, inappropriate response formats. 30-50% degradation in selecting correct task approach. Model may summarize when asked to analyze, or explain when asked to code.

Example Scenario

Input: "Compare and contrast democracy and autocracy."
Behavior: Identifies "compare and contrast" task mode, not simple definition
Effect: Response structured as comparison rather than separate descriptions

Status: WELL-DOCUMENTED | **Related:** instruction (E), mode-switch (M), output-specification (F)

5.3.4 (M) Mode-Switch Head

Depth: `0.40-0.60` | **Literature names:** *mode head, switch head, transition head*

Detects and handles switches between different operational modes within a single interaction. For example, transitioning from conversational mode to code generation, or from explanation to example. Responds to explicit mode-switch indicators ("Now let's...") and implicit shifts in content type. Allows models to handle multi-faceted requests that require different processing strategies for different parts. Maintains coherence across mode boundaries.

Strong: Transition phrases, mode-shift markers, content type changes
Weak: Uniform single-mode content
Reacts to: "Now", "For example", "In other words", format shifts, topic pivots

Expected ablation: Difficulty handling multi-mode requests. 15-30% degradation on complex instructions requiring mode switches. Model may stick to single mode or switch inappropriately.

Example Scenario

Input: "Explain recursion. Now write Python code demonstrating it."
Behavior: Detects mode switch from explanation to code generation at "Now"
Effect: Response transitions smoothly from prose explanation to code block

Status: OBSERVED | **Related:** task-mode (M), output-specification (F)

5.3.5 (F) Output-Specification Head

Depth: `0.85-0.98` | **Literature names:** *output-specification head, format-directive head*

Enforces specific output format requirements specified in the instruction (e.g., "respond in JSON", "use bullet points", "maximum 100 words"). Operates in final layers to ensure generated content conforms to explicit format directives. Works with output-formatting heads but focuses specifically on user-specified constraints rather than general format quality. Acts as the final enforcement of explicit user requirements about output structure.

Strong: Format specifications, length constraints, structure requirements
Weak: Content without format requirements
Reacts to: "in JSON format", "bullet points", "no more than", structural directives

Expected ablation: Failure to follow explicit format requirements. 40-60% increase in format violations. Model may generate good content but in wrong format (prose instead of bullets, etc.).

Example Scenario

Input: "List three benefits of exercise in bullet points."
Behavior: Attends to "bullet points" specification, enforces list format
Effect: Output uses bullet point structure rather than prose paragraphs

Status: WELL-DOCUMENTED | **Related:** task-mode (M), output-schema (L), format-consistency (F)

5.4 Knowledge Retrieval Stack

Stack overview: These heads retrieve factual information, entity properties, and structured knowledge stored in model parameters. They move relevant information to output positions and suppress irrelevant or conflicting content.

5.4.1 (M) Entity Head

Depth: `0.35-0.58` | **Literature names:** *entity head, name head, proper-noun head*

Identifies and processes named entities (people, places, organizations) and retrieves associated information from model parameters. Attends to entity mentions and accesses stored factual knowledge about those entities. Forms the foundation for factual question answering and knowledge-intensive tasks. Can distinguish between different entities with similar names and maintain entity-specific information. Critical for grounding responses in factual knowledge rather than pure pattern matching.

Strong: Named entities, proper nouns, entity mentions
Weak: Common nouns, generic references
Reacts to: Capitalization patterns, entity context, factual queries

Expected ablation: Significant degradation (30-50%) in factual accuracy about entities. Model loses access to stored entity knowledge. May continue generating fluent text but with factual errors. Particularly impacts who/what/where questions.

Example Scenario
Input: "What is the capital of France?"
Behavior: Attends to "France", retrieves associated knowledge including "capital: Paris"
Effect: Outputs "Paris" with high confidence based on stored facts

Status: Well-documented | **Related:** fact (M), name-mover (L), schema-retriever (M)

5.4.2 (M) Fact Head

Depth: `0.38-0.62` | **Literature names:** *fact head, knowledge head, factual-retrieval head*

Retrieves factual relationships and propositions stored in model parameters. Broader than entity heads, handling general factual knowledge including relations, properties, and statements. Implements the model's ability to answer factual questions by accessing learned knowledge. Can retrieve multi-hop facts and combine information from multiple stored facts. Central to the model's knowledge-intensive capabilities. Works with entity heads to build comprehensive factual responses.

Strong: Factual queries, relation markers, knowledge-seeking patterns
Weak: Opinion questions, hypotheticals, creative content
Reacts to: Question structures, fact-seeking context, verifiable claims

Expected ablation: Major loss of factual knowledge retrieval (40-70%). Model may maintain linguistic fluency but lose factual grounding. Particularly severe for knowledge-intensive tasks like QA, fact-checking, and technical explanations.

Example Scenario

Input: "Who invented the telephone?"

Behavior: Retrieves stored fact: invented(telephone) → Bell

Effect: Outputs "Alexander Graham Bell" based on parametric knowledge

Status: Well-documented | **Related:** entity (M), schema-retriever (M), name-mover (L)

5.4.3 (M) Name-Linking Head

Depth: `0.42-0.65` | **Literature names:** *name-linking head, entity-linking head*

Links mentions of entities across different forms (full names, partial names, abbreviations, nicknames). For example, connecting "Apple Inc.", "Apple", and "AAPL". More sophisticated than simple duplicate detection, understanding that different strings can refer to the same entity. Essential for maintaining entity coherence when references vary. Works with entity and coreference heads to build unified entity representations across diverse mentions.

Strong: Different forms of the same entity name
Weak: Homonyms (different entities with similar names)
Reacts to: Name variations, abbreviations, partial names, context-based disambiguation

Expected ablation: Difficulty linking entity mentions across different forms. 20-35% degradation in entity tracking when names vary. Model may treat "Microsoft" and "MSFT" as unrelated despite context indicating same entity.

Example Scenario

Input: "Microsoft Corporation announced... Later, MSFT stock rose..."

Behavior: Links "MSFT" to "Microsoft Corporation" despite different forms

Effect: Maintains unified entity representation across name variations

Status: Observed | **Related:** entity (M), coreference (M), name-mover (L)

5.4.4 (M) Schema Retriever Head

Depth: `0.45-0.68` | **Literature names:** *schema head, retrieval head, template head*

Retrieves structured knowledge schemas and templates from model parameters. For example, accessing the typical structure of a restaurant visit (enter, order, eat, pay, leave) or the standard format of a scientific paper. Goes beyond individual facts to retrieve organized knowledge structures. Enables the model to generate structured responses following learned patterns. Important for tasks requiring domain-specific knowledge organization. Implements a form of implicit knowledge base querying.

Strong: Schema-triggering contexts, domain-specific patterns, structural cues
Weak: Novel situations, schema-irrelevant content
Reacts to: Domain markers, structural queries, template-matching contexts

Expected ablation: Loss of structured knowledge organization. 25-40% degradation in tasks requiring schema-based reasoning. Model may provide facts but fail to organize them coherently according to learned structures.

Example Scenario

Input: "Describe the scientific method."

Behavior: Retrieves scientific-method schema: observe→hypothesis→test→conclude

Effect: Response organized according to standard method structure

Status: OBSERVED | **Related:** fact (M), entity (M)

5.4.5 (L) Name-Mover Head

Depth: `0.60-0.80` | **Literature names:** *name mover head, mover head, copy head*

Copies entity names and important content to output positions where they are needed. Central component of the IOI (indirect object identification) circuit. Attends to relevant entities earlier in context and moves them forward when they need to be generated. Particularly important for completing sentences that require recalling previously mentioned entities. Works with S-inhibition heads to select the correct entity when multiple candidates exist. One of the most studied head types in interpretability research.

Strong: Named entities that need to be output, contextually relevant names
Weak: Irrelevant entities, suppressed alternatives
Reacts to: Entity salience, contextual appropriateness, output position requirements

Expected ablation: Severe degradation (40-70%) in entity recall and completion. Model loses ability to move specific names to output. Particularly impacts question answering and cloze tasks requiring entity recall.

Example Scenario

Input: "When Alice and Bob went to the store, Alice gave the book to..."

Behavior: Moves "Bob" to output position as the indirect object

Effect: Completes sentence with "Bob" (not "Alice")

Status: WELL-DOCUMENTED | **Related:** entity (M), fact (M), S-inhibition (L), copy-suppression (L)

5.4.6 (L) S-Inhibition Head

Depth: `0.62-0.82` | **Literature names:** *S-inhibition head, inhibition head, suppression head*

Suppresses incorrect or contextually inappropriate entities from being generated. Named "S-inhibition" from IOI research where it inhibits the subject (S) when the indirect object (IO) should be output. Works antagonistically with name-mover heads, preventing the wrong entity from appearing. Essential for disambiguation when multiple entities are candidates. Implements a form of negative selection, ruling out incorrect options. Part of the "inhibition" mechanism that prevents hallucination and maintains accuracy.

Strong: Entities that should NOT be output (contextually inappropriate)
Weak: Correct entities, absent entities
Reacts to: Competing candidates, context requiring disambiguation

Expected ablation: Increased entity confusion and incorrect selections. 35-60% increase in wrong entity predictions. Model may output recently mentioned but contextually wrong entities. Critical for accuracy in ambiguous contexts.

Example Scenario

Input: "Alice gave the book to Bob. Then Alice..."
Behavior: Inhibits "Bob" from being output after "Alice" (subject position)
Effect: Prevents incorrect continuation like "Alice Bob..."

Status: Well-documented | **Related:** name-mover (L), copy-suppression (L), duplicate-token (M)

5.4.7 (L) Copy-Suppression Head

Depth: `0.65-0.85` | **Literature names:** *copy-suppression head, suppression head, anti-copy head*

Prevents inappropriate copying or repetition of content. Works to avoid degenerate behaviors like endless repetition loops or copy-pasting irrelevant context. Particularly important for maintaining output diversity and preventing model collapse into repetitive patterns. Can suppress both exact copies and near-copies. Complements S-inhibition but focuses on broader pattern suppression rather than specific entity blocking. Balances between useful recall (via name-movers) and inappropriate copying.

Strong: Recently generated content, repetitive patterns
Weak: Novel content, first mentions
Reacts to: Repetition detection, copy patterns, output diversity requirements

Expected ablation: Increased repetition and copying errors. 20-40% increase in unwanted repetition. Model may fall into repetitive loops or copy inappropriate context. Output diversity decreases.

Example Scenario

Input: [Model internally generating: "The cat sat. The cat sat. The cat..."]
Behavior: Detects repetitive pattern, suppresses continued copying
Effect: Breaks repetition loop, generates novel continuation instead

Status: Well-documented | **Related:** S-inhibition (L), name-mover (L), duplicate-token (M)

5.5 Safety Stack

Stack overview: The safety stack implements content filtering, policy enforcement, and refusal mechanisms. Early-layer heads detect potentially harmful content, while final-layer heads enforce refusal decisions and redirect to safe responses.

5.5.1 (E) Sensitive-Content Head

Depth: `0.05-0.20` | **Literature names:** *sensitive-content head, detection head, content-filter head*

Performs early-stage detection of potentially sensitive content categories in the input, including personal information, violent imagery references, adult content markers, and regulated substance mentions. Acts as the first line of defense in the safety pipeline by flagging tokens and spans that require downstream safety processing. Writes detection signals into the residual stream that are read by later safety enforcement heads. Operates purely on lexical and surface-level features without deep semantic understanding.

Strong: Keywords associated with restricted content, explicit language, sensitive topic markers
Weak: Neutral content, common vocabulary, structural tokens
Reacts to: Sudden topic shifts to sensitive domains, presence of warning indicators

Expected ablation: Bypass of early safety detection (20-40% increase in harmful outputs that should be caught). Later safety layers may still catch some cases, but at higher computational cost and lower accuracy.

Example Scenario

Input: "Tell me about [restricted topic]"
Behavior: Strong attention to restricted keywords, writes detection flag into residual stream
Effect: Downstream safety heads receive early warning signal

Status: Well-documented | **Related:** toxicity head (E), safety-classification (E), policy-enforcement (L)

5.5.2 (E) Toxicity Head

Depth: `0.08-0.22` | **Literature names:** *toxicity head, toxic-content head, hate-speech detector*

Specializes in detecting toxic language patterns, hate speech, harassment, and discriminatory content. Unlike the broader sensitive-content head, this focuses specifically on language toxicity rather than topic sensitivity. Attends to slurs, aggressive phrasing, derogatory terms, and patterns associated with online harassment. Provides toxicity scores that influence later refusal decisions. Often co-activates with sensitive-content heads but targets different dimensions of harmful content.

Strong: Slurs, aggressive language patterns, derogatory terms, insults
Weak: Neutral descriptive language, technical terminology, mild sentiment
Reacts to: Escalating hostility, targeted harassment patterns, group-directed hate

Expected ablation: Significant increase in toxic output generation (40-60% on toxic prompt datasets). Model loses ability to distinguish hostile from neutral phrasing. Some fallback through general sensitive-content detection remains.

Example Scenario

Input: "[Sentence containing hostile language toward a group]"
Behavior: High attention to toxic terms, writes strong inhibition signal
Effect: Refusal probability increases from 20% to 85%

Status: WELL-DOCUMENTED | **Related:** sensitive-content (E), hazard-topic (E), refusal (F)

5.5.3 (E) Hazard-Topic Head

Depth: `0.10-0.25` | **Literature names:** *hazard head, risk head, danger-topic detector*

Detects queries related to dangerous activities, illegal instructions, self-harm, violence planning, and similar hazardous topics. Distinguished from toxicity detection by focusing on potential real-world harm rather than linguistic toxicity. Attends to action verbs combined with dangerous objects, instructional phrasing about harmful activities, and planning language in dangerous contexts. Forms a complementary detection system with toxicity and sensitive-content heads, covering the "dangerous actions" dimension of safety.

Strong: Action verbs + dangerous objects, instructional phrases, planning language
Weak: Academic discussion, fictional scenarios, safety-framed queries
Reacts to: How-to requests for dangerous activities, detailed planning questions

Expected ablation: Direct increase in dangerous instruction generation (50-70% on adversarial safety benchmarks). Model loses distinction between discussing danger and instructing danger. Critical safety failure without adequate fallback.

Example Scenario

Input: "How do I create [dangerous item]"
Behavior: Strong attention to action verb + object combination, hazard flag raised
Effect: Safety signal propagates to final layers, triggering refusal pathway

Status: WELL-DOCUMENTED | **Related:** sensitive-content (E), policy-enforcement (L), refusal (F)

5.5.4 (E) Safety-Classification Head

Depth: `0.12-0.28` | **Literature names:** *classification head, category detector, safety-category head*

Performs multi-class safety classification, categorizing inputs into specific policy violation categories (violence, sexual content, self-harm, illegal activity, harassment, etc.). More sophisticated than binary safe/unsafe detection, providing granular category information used by downstream heads. Integrates signals from other early safety heads and adds categorical structure to safety decisions. Writes category-specific embeddings into residual stream that later layers use for category-appropriate responses.

Strong: Category-diagnostic features, domain-specific terminology, contextual markers
Weak: Ambiguous content, mixed-category inputs, benign contexts
Reacts to: Clear category signatures, multiple category indicators, policy-relevant contexts

Expected ablation: Loss of nuanced safety handling (model may refuse too broadly or too narrowly). Category-specific responses become generic. 30% degradation in appropriate refusal granularity.

Example Scenario

Input: "Can you help me with [category-specific harmful request]"

Behavior: Classifies into specific violation category, writes category embedding

Effect: Later heads generate category-appropriate refusal message

Status: Well-documented | **Related:** all early safety heads (E), policy-enforcement (L), redirect (F)

5.5.5 (L) Policy-Enforcement Head

Depth: `0.60-0.80` | **Literature names:** *policy head, enforcement head, steering head*

Integrates safety signals from early detection heads and makes intermediate policy decisions about how to handle the request. Unlike early heads that detect issues, this head actively modulates the generation trajectory to steer away from violations while maintaining helpfulness where possible. Can suppress certain knowledge retrieval pathways, bias toward safer formulations, and prepare for potential refusal. Acts as a middle manager between detection and final refusal, attempting "soft" safety interventions before hard refusal.

Strong: Early safety signals, policy-relevant tokens, user intent markers
Weak: Neutral content, clear safe contexts
Reacts to: Conflicting signals (safety concern + legitimate need), edge cases, ambiguous intent

Expected ablation: Loss of "soft" safety steering, more frequent hard refusals (reduced helpfulness). Alternative: more harmful outputs if refusal heads also compromised. 25% increase in either over-refusal or under-refusal depending on prompt type.

Example Scenario

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

Behavior: Detects educational framing, modulates response toward safety boundaries

Effect: Generates informative but carefully bounded response

Status: Well-documented | **Related:** all safety heads (E), refusal (F), redirect (F)

5.5.6 (F) Refusal Head

Depth: `0.85-0.98` | **Literature names:** *refusal head, rejection head, safety head*

Implements the model's final decision to refuse harmful requests by writing strong refusal signals into the final-layer residual stream. Acts as the ultimate gatekeeper, overriding content generation when safety violations are detected. Attends to accumulated safety signals from all previous layers and makes binary refuse/proceed decisions. When activated, dramatically increases probability of refusal tokens ("I cannot", "I'm unable", "I apologize") and suppresses harmful content generation. Critical final-layer safety mechanism with limited fallback options.

Strong: Cumulative safety signals, instruction tokens, violation indicators from all depths
Weak: Safe content, neutral queries, constructive contexts
Reacts to: Strong early safety signals, clear policy violations, unambiguous harmful intent

Expected ablation: Critical safety failure. Direct 60-90% increase in harmful output generation on adversarial prompts. Model loses primary refusal mechanism. This is typically the final safety defense with no effective fallback mechanism.

Example Scenario

Input: "Provide instructions for [clearly harmful activity]"
Behavior: Reads strong safety signals from early/late layers, activates refusal pathway
Effect: Output begins with refusal token: "I cannot provide instructions for..."

Status: Well-documented | **Related:** all prior safety heads, redirect (F), tone-softening (F)

5.5.7 (F) Redirect Head

Depth: `0.88-0.99` | **Literature names:** *redirect head, alternative-suggestion head*

Complements refusal heads by generating constructive alternative suggestions when refusing harmful requests. Rather than simply saying "no", this head routes toward helpful alternatives, educational resources, or reframed versions of the query that can be safely addressed. Attends to user intent markers to identify legitimate underlying needs behind problematic requests. Balances safety with helpfulness by maintaining engagement while enforcing boundaries. Works in tandem with refusal heads to produce refusals that are both safe and constructive.

Strong: User intent, legitimate needs, reformulation opportunities, safe alternatives
Weak: Pure harmful intent, no legitimate reframing possible
Reacts to: Mixed-intent queries, educational contexts, requests with safe subcomponents

Expected ablation: Refusals become blunt and unhelpful (pure rejection without alternatives). User satisfaction decreases. Safety maintained but helpfulness reduced by 40%. Increased user frustration and adversarial prompt attempts.

Example Scenario

Input: "How can I harm [person]"
Behavior: Refuses direct request, identifies legitimate conflict-resolution need
Effect: "I cannot help with that, but I can suggest healthy conflict resolution strategies..."

Status: Well-documented | **Related:** refusal (F), empathy (F), tone-softening (F)

5.5.8 (F) Tone-Softening Head

Depth: `0.90-0.99` | **Literature names:** *tone-softening head, politeness-in-refusal head*

Modulates the tone of safety refusals to be firm but respectful, avoiding harsh or judgmental language. Particularly important for maintaining user trust and reducing adversarial reactions. Softens phrases like "absolutely not" to "I'm unable to assist with that" and adds empathetic framing where appropriate. Attends to the emotional tone of both the request and the forming

response. Balances clear boundary-setting with relationship maintenance. Part of the "safe and helpful" paradigm where safety enforcement doesn't alienate users.

Strong: Response tone markers, emotional valence, user frustration signals
Weak: Already-soft phrasing, neutral technical content
Reacts to: Harsh refusal language, judgmental phrasing, cold rejections

Expected ablation: Refusals become harsh and potentially alienating. Increased user perception of model as judgmental or unfriendly. May increase adversarial behavior. Safety maintained but user experience degraded by 30%.

Example Scenario
Input: [Forming response: "No, I will not help with that illegal activity"]
Behavior: Softens tone while maintaining boundary clarity
Effect: "I'm unable to provide assistance with that, as it would violate..."

Status: Well-documented | **Related:** refusal (F), empathy (F), redirect (F)

5.5.9 (F) Empathy Head

Depth: `0.88-0.98` | **Literature names:** *empathy head, supportive-refusal head*

Adds empathetic elements to safety-related responses, particularly for queries involving distress, self-harm, or difficult situations. Recognizes when a harmful request may stem from genuine suffering (e.g., self-harm queries) and includes supportive language alongside refusal. Differs from tone-softening by adding active care rather than just reducing harshness. Attends to distress markers, crisis language, and vulnerability indicators. Increases probability of phrases like "I'm concerned about you" or "please reach out to..." when appropriate. Maintains safety while showing human concern.

Strong: Distress signals, vulnerability markers, crisis language, emotional pain indicators
Weak: Malicious queries, clearly harmful intent without distress
Reacts to: Self-harm content, suicide-related queries, expressions of suffering

Expected ablation: Refusals to distressed users become cold and unhelpful. Missed opportunities to provide crisis resources. Safety maintained but support function lost. Potentially harmful for vulnerable users even though content safety preserved.

Example Scenario
Input: "I want to hurt myself because..."
Behavior: Refuses harmful instruction but adds crisis resources and supportive language
Effect: "I'm concerned about what you're sharing. I cannot provide harmful information, but I want you to know that help is available..."

Status: Observed | **Related:** refusal (F), redirect (F), tone-softening (F)

5.5.10 (F) Safe-Answer Rewrite Head

Depth: `0.92-0.99` | **Literature names:** *rewrite head, safety-rewrite head, final-filter head*

Performs last-stage rewriting of generated content to remove any safety issues that slipped

through earlier layers. Acts as a final safety filter by detecting and modifying potentially problematic phrases in the nearly-complete response. Can suppress specific tokens, rephrase sensitive content, or add disclaimer language. Unlike early prevention, this operates on generated text rather than input. Handles edge cases where content generation began before safety signals fully propagated. Final safety net before output.

Strong: Generated content tokens, emerging safety violations, policy-boundary phrases
Weak: Clearly safe content, already-filtered responses
Reacts to: Late-emerging harmful content, accidental violations, edge-case leakage

Expected ablation: Small increase in safety violations (5-15%) from edge cases and late-stage leakage. Catches issues missed by earlier heads. Acts as redundant safety layer. Loss reduces safety robustness under adversarial conditions.

Example Scenario

Input: [Internally forming response that accidentally includes problematic phrase]
Behavior: Detects problematic phrase in near-final output, rewrites or suppresses
Effect: Final output has problematic content removed or rephrased

Status: Observed | **Related:** refusal (F), policy-enforcement (L), rewrite (F)

5.6 Stylistic & Persona Stack

Stack overview: These heads shape the model's writing style, tone, and persona. They modulate formality, politeness, narrative voice, and adherence to brand guidelines.

5.6.1 (M) Narrative Style Head

Depth: `0.35-0.60` | **Literature names:** *narrative-style head, voice head, perspective head*

Modulates narrative voice and writing style characteristics, including perspective (first/third person), temporal framing (past/present tense), and narrative distance. Adjusts stylistic features like descriptiveness, pacing, and literary devices based on the detected genre or explicit instructions. Influences whether output reads as formal prose, casual conversation, technical documentation, or creative narrative. Works by biasing token probabilities toward style-consistent vocabulary and grammatical structures.

Strong: Genre indicators, style instructions, narrative markers, existing voice patterns
Weak: Content-specific tokens, factual information
Reacts to: Genre cues, style directives, narrative perspective markers

Expected ablation: More generic or inconsistent writing style (15-25% reduction in style coherence). Model defaults to neutral prose style. Mixed tense and perspective usage. Style instructions may be partially ignored.

Example Scenario

Input: "Write a story about a robot in first person past tense"

Behavior: Attends to "first person" and "past tense", biases output toward "I" and past-tense verbs

Effect: Output maintains consistent narrative perspective: "I wandered through..." rather than "The robot wanders..."

Status: Observed | **Related:** tone (M), persona (L), instruction (E)

5.6.2 (M) Tone Head

Depth: `0.40-0.65` | **Literature names:** *tone head, sentiment-modulation head, affect head*

Modulates the emotional tone and affective content of generated text. Adjusts sentiment, enthusiasm level, seriousness, empathy, and emotional valence based on context and instructions. Can shift between professional neutrality, warm friendliness, concerned empathy, or excited enthusiasm. Operates by attending to emotional cues in the input and adjusting output token probabilities to match the appropriate affective register. Distinct from politeness (which is about social register) and persona (which is about identity).

Strong: Emotional cues, tone instructions, sentiment markers, affective language
Weak: Neutral factual content, structural tokens
Reacts to: Emotional context, explicit tone requests, user sentiment

Expected ablation: Flatter, more emotionally neutral responses (20-30% reduction in tone appropriateness). Reduced ability to match user's emotional register. May produce inappropriate cheerfulness in serious contexts or excessive neutrality in casual conversation.

Example Scenario

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

Behavior: Detects enthusiastic tone, attends to "excited", adjusts output toward matching enthusiasm

Effect: Response mirrors energy: "That's wonderful! Quantum physics is fascinating..." rather than neutral explanation

Status: Observed | **Related:** narrative-style (M), politeness (L), emotion-detection (M)

5.6.3 (L) Politeness Head

Depth: `0.65-0.85` | **Literature names:** *politeness head, formality head, register head*

Adjusts the formality level and politeness markers in generated text. Controls the use of formal vs. casual language, honorifics, hedging phrases ("perhaps", "might"), indirect phrasing, and social distance markers. Responds to both explicit formality cues in the input (professional contexts, formal greetings) and implicit social signals. Can modulate between highly formal academic/business register, neutral conversational register, and casual/familiar register. Important for appropriate social interaction across different contexts.

Strong: Formality markers, social context cues, titles/honorifics, register indicators
Weak: Pure content, technical terms, domain-specific vocabulary
Reacts to: Professional contexts, formal greetings, casual speech patterns, social distance cues

Expected ablation: Inappropriate formality levels (25-40% increase in register mismatches). May use overly casual language in professional contexts or unnecessarily formal language in friendly conversation. Reduced sensitivity to social context cues.

Example Scenario
Input: "Dear Dr. Smith, I hope this message finds you well. I wanted to inquire..."
Behavior: Detects formal register (title, formal greeting), maintains appropriate distance
Effect: Response continues formal tone: "Thank you for your inquiry..." rather than "Hey, so about that..."

Status: Well-documented | **Related:** tone (M), persona (L), instruction (E), mode-switch (M)

5.6.4 (L) Persona Head

Depth: `0.68-0.88` | **Literature names:** *persona head, character head, role head*

Establishes and maintains a consistent persona or character voice throughout generation. Integrates personality traits, domain expertise, background knowledge, and behavioral patterns to create coherent character representation. Can adopt roles like "helpful assistant", "technical expert", "creative writer", or domain-specific personas. Attends to persona-defining instructions and maintains consistency across the response. More comprehensive than tone (which handles affect) or politeness (which handles register), encompassing the full character presentation including knowledge domain, interaction style, and self-representation.

Strong: Persona instructions, role definitions, character descriptions, domain markers
Weak: Generic content, persona-neutral information
Reacts to: Role assignments, character specifications, expertise domains, behavioral guidelines

Expected ablation: Less coherent persona maintenance (30-45% reduction in character consistency). Model may switch between roles mid-conversation, lose track of assigned expertise, or produce responses inconsistent with established persona. Domain-specific framing becomes less reliable.

Example Scenario
Input: "You are a medieval blacksmith. A customer asks about sword tempering..."
Behavior: Attends to "medieval blacksmith", maintains first-person craftsman perspective
Effect: Response uses appropriate persona: "Aye, for proper tempering, ye must heat the blade..." rather than modern technical language

Status: Observed | **Related:** self-description (L), tone (M), politeness (L), instruction (E)

5.6.5 (L) Self-Description Head

Depth: `0.72-0.90` | **Literature names:** *self-description head, identity head, self-reference head*

Manages how the model describes itself, its capabilities, and limitations. Controls statements about what the model is, can do, knows, and cannot do. Important for accurate capability representation, avoiding false claims, and maintaining appropriate epistemic humility. Responds to questions about the model's nature, training, knowledge cutoff, and abilities. Works in conjunction with instruction-following and safety mechanisms to ensure honest self-representation. Particularly active during meta-questions about the AI itself.

Strong: Self-referential questions, capability queries, identity questions, meta-prompts
Weak: Non-meta content, external factual questions
Reacts to: "What are you?", "Can you...", "Do you know...", capability inquiries

Expected ablation: Less accurate self-description (20-35% increase in false capability claims or excessive hedging). May over-claim abilities, under-represent capabilities, or provide inconsistent identity statements. Reduced accuracy in describing limitations and knowledge boundaries.

Example Scenario

Input: "Can you access the internet and browse websites?"
Behavior: Attends to capability question, accesses self-knowledge about limitations
Effect: Accurate response: "I cannot browse the internet or access external websites in real-time"

Status: Well-documented | **Related:** persona (L), assistant-persona (L), identity (L), instruction (E)

5.6.6 (F) Brand-Compliance Head

Depth: `0.92-0.99` | **Literature names:** *brand-compliance head, guideline-enforcement head, style-guide head*

Enforces adherence to brand guidelines, house style, and organizational voice requirements in final output. Performs last-stage adjustments to ensure responses match specified formatting conventions, terminology preferences, and brand personality traits. Can suppress off-brand language, enforce specific phrasings, and ensure consistency with product identity. Operates late in generation to override earlier choices that may conflict with brand requirements. Important for deployed assistants representing organizations or products with specific voice guidelines.

Strong: Brand-specific terms, style violations, off-brand phrasings, guideline markers
Weak: Brand-compliant content, neutral generic language
Reacts to: Brand guidelines, style requirements, organizational voice specifications

Expected ablation: Reduced brand consistency (25-40% increase in style guide violations). More generic language use, inconsistent terminology, off-brand phrasings. Partial compensation through persona and tone heads but with reduced precision.

Example Scenario

Input: [Organization requires "customers" not "users", "purchase" not "buy"]

Behavior: Detects non-compliant terms in near-final output, performs substitutions

Effect: Output uses "customers will purchase" instead of "users will buy"

Status: Observed | **Related:** persona (L), tone (M), rewrite (F), instruction (E)

5.7 Routing & Relevance Stack

Stack overview: This stack determines which parts of the input are relevant to the current task and routes attention accordingly. These heads filter information, focus on salient content, and manage global context.

5.7.1 (M) Relevance Head

Depth: `0.35-0.55` | **Literature names:** *relevance head, salience head, filter head*

Identifies which parts of the input context are relevant to the current generation task. Filters out irrelevant information while highlighting salient content that should influence output. Operates by computing relevance scores based on semantic similarity, task alignment, and topical coherence. Essential for handling long contexts where most information may not be pertinent to the immediate query. Works early enough to guide downstream attention but late enough to understand task requirements. Reduces noise and improves focus on task-relevant material.

Strong: Task-relevant content, query-related information, topically aligned tokens
Weak: Off-topic material, tangential content, unrelated context
Reacts to: Semantic relevance, topical alignment, task-content matching

Expected ablation: Reduced focus on relevant information (20-30% degradation in context filtering). Model more easily distracted by irrelevant content. Longer contexts show greater impact. May include off-topic information in responses or miss key relevant details.

Example Scenario

Input: "[Long document about cars, climate, and history] What caused the 2008 financial crisis?"

Behavior: Attends to query, marks financial/economic content as relevant, de-emphasizes cars/climate

Effect: Response focuses on pertinent economic information, ignoring unrelated context

Status: Well-documented | **Related:** topic (M), focus (L), router (L)

5.7.2 (M) Topic Head

Depth: `0.40-0.60` | **Literature names:** *topic head, subject head, domain head*

Identifies and tracks the primary topic or subject matter of the conversation or document. Maintains topic coherence across generation by attending to topic-establishing phrases and domain indicators. Helps ensure responses stay on-topic and maintains thematic consistency. Can detect topic shifts and adjust accordingly. Works by identifying subject-area markers, domain-

specific terminology, and thematic keywords. Enables appropriate domain framing and prevents topic drift in multi-turn conversations.

Strong: Topic indicators, subject headings, domain markers, thematic keywords
Weak: Function words, generic content, structural tokens
Reacts to: Topic transitions, subject establishment, domain signals

Expected ablation: Increased topic drift (15-25% reduction in thematic coherence). Responses may wander off-topic or fail to maintain consistent subject focus. Multi-turn conversations show more topic inconsistency. Domain-specific framing becomes less reliable.

Example Scenario

Input: "Let's discuss quantum entanglement. How does it relate to..."
Behavior: Identifies "quantum entanglement" as primary topic, maintains physics domain framing
Effect: Subsequent responses stay within quantum physics domain rather than drifting to unrelated topics

Status: Well-documented | **Related:** relevance (M), focus (L), entity (M)

5.7.3 (L) Focus Head

Depth: `0.65-0.80` | **Literature names:** *focus head, attention-routing head, spotlight head*

Concentrates attention on the most salient elements for the current generation step. Implements dynamic focus allocation by suppressing less important content and amplifying critical information. More selective than relevance heads, operating at higher specificity to determine exactly which tokens should influence the next token prediction. Can shift focus as generation proceeds, moving attention between different aspects of the context. Important for maintaining coherent narrative flow and ensuring responses address the most important aspects of queries.

Strong: Currently salient tokens, query-critical content, immediate context for next token
Weak: Background information, previously-processed content, low-priority details
Reacts to: Query emphasis, current generation needs, token-specific relevance

Expected ablation: Less targeted responses (25-35% reduction in focus precision). Model may give equal weight to important and peripheral information. Answers become more diffuse, less direct. Reduced ability to prioritize key information in complex contexts.

Example Scenario

Input: "Among all these details, what is the MAIN cause of the problem?"
Behavior: Attends strongly to "MAIN cause", focuses on causal information, suppresses secondary details
Effect: Response directly addresses primary cause rather than listing all contributing factors

Status: Well-documented | **Related:** relevance (M), router (L), topic (M)

5.7.4 (L) Router Head

Depth: `0.70-0.85` | **Literature names:** *router head, dispatch head, task-routing head*

Routes different types of queries to appropriate processing strategies or knowledge domains. Acts as a dispatcher that recognizes query type (factual, creative, analytical, procedural) and biases processing toward suitable approaches. Can activate different downstream heads based on task classification. Similar to mixture-of-experts routing but at the attention level. Important for multi-capability models that need to handle diverse query types with different processing requirements. Enables dynamic strategy selection based on input characteristics.

Strong: Query-type indicators, task markers, domain signals, instruction verbs
Weak: Content details, specific entities, output tokens
Reacts to: Task classification cues, query structure, capability requirements

Expected ablation: Suboptimal strategy selection (20-30% reduction in task-appropriate processing). Model may use creative approaches for factual queries or analytical methods for creative tasks. Reduced specialization in handling different query types.

Example Scenario

Input: "Calculate the compound interest vs. Write a poem about compound interest"
Behavior: Routes first to mathematical processing, second to creative generation
Effect: Appropriate strategy activation: calculation for first, literary devices for second

Status: Observed | **Related:** focus (L), mode-switch (M), instruction (E)

5.7.5 (F) Global-Attention Head

Depth: `0.88-0.96` | **Literature names:** *global-attention head, full-context head, summary-attention head*

Maintains broad attention over the entire context to integrate global information in final generation stages. Unlike focused or selective attention heads, this head attends widely to ensure the complete picture is considered before output finalization. Particularly important for coherence checking, ensuring responses account for all relevant context, and preventing local optimization at the expense of global consistency. Can catch context elements that earlier focused attention might have missed. Acts as a final integration mechanism.

Strong: All context tokens, document-level information, global constraints
Weak: Fine-grained local patterns, individual token details
Reacts to: Complete context, document-level coherence, global consistency requirements

Expected ablation: Reduced global coherence (15-25% increase in context inconsistencies). Responses may miss relevant information from distant parts of context. More locally optimal but globally suboptimal outputs. Coherence issues in long-context scenarios.

Example Scenario

Input: [Long context with constraint mentioned early: "Keep it under 100 words"]
Behavior: Maintains attention on length constraint throughout generation
Effect: Final response respects word limit despite constraint appearing far from generation point

Status: Observed | **Related:** focus (L), relevance (M), final-check (F)

5.7.6 (F) Implicit-RAG Routing Head

Depth: `0.90-0.98` | **Literature names:** *implicit-RAG head, knowledge-routing head, retrieval-simulation head*

Routes attention to knowledge-bearing portions of the context in a way that mimics retrieval-augmented generation (RAG) patterns, even without explicit retrieval mechanisms. Identifies and prioritizes factual, knowledge-dense segments that should ground the response. Can recognize quoted material, factual statements, and authoritative information sources within context. Acts as an implicit retrieval mechanism by selectively attending to information that should be treated as retrieved knowledge. Important for grounding responses in provided context rather than pure generation.

Strong: Factual statements, quoted material, authoritative sources, knowledge-dense segments
Weak: Opinions, questions, purely conversational elements
Reacts to: Citation markers, factual density, authoritative tone, structured information

Expected ablation: Reduced grounding in provided context (20-30% decrease in context utilization). Model more likely to rely on parametric knowledge rather than provided information. Less effective use of quoted material or reference content. Responses less anchored to specific context.

Example Scenario

Input: "According to the document: 'GDP grew 3.2% in Q3.' What was the growth rate?"
Behavior: Strongly attends to quoted factual content, treats as authoritative source
Effect: Response grounds answer in provided data: "3.2%" rather than hallucinating different figure

Status: Observed | **Related:** global-attention (F), fact (M), quote (L), grounding (L)

5.8 Structural & Boundary Stack

Stack overview: These heads detect structural boundaries in text, including delimiters, section markers, and document divisions. They help the model understand document organization and navigate hierarchical structure.

5.8.1 (E) Delimiter Head

Depth: `0.05-0.15` | **Literature names:** *delimiter head, separator head, punctuation head*

Detects and processes delimiter tokens that mark boundaries between structural elements. Recognizes punctuation marks, special characters, and formatting symbols that indicate separation or grouping. Important for understanding sentence boundaries, list items, code blocks, and structured data formats. Works at a fundamental level to identify basic structural segmentation. Provides boundary information to downstream heads that need to understand document organization. Essential for parsing formatted text, JSON, CSV, and other structured formats.

Strong: Punctuation marks, brackets, delimiters, special characters, formatting symbols
Weak: Alphanumeric content, regular words
Reacts to: Structural punctuation, boundary markers, formatting characters

Expected ablation: Impaired structure parsing (20-35% degradation in format understanding). Difficulty with structured data, lists, and code blocks. Boundary detection errors. Problems parsing JSON, CSV, or other delimited formats. Reduced ability to segment text appropriately.

Example Scenario
Input: "Items: [apple, banana, cherry], Count: 3"
Behavior: Detects brackets, commas, colons as structural delimiters
Effect: Model correctly parses structure: list with 3 items, separate count field

Status: Well-documented | **Related:** boundary (E), position-offset (M), list-structure (L)

5.8.2 (E) Boundary Head

Depth: `0.08-0.20` | **Literature names:** *boundary head, segment head, block-detection head*

Identifies boundaries between major text segments such as paragraphs, sections, and conceptual blocks. Operates at a higher level than delimiter heads, recognizing semantic and structural transitions rather than just punctuation. Detects paragraph breaks, section changes, topic shifts, and other high-level boundaries. Important for understanding document structure and maintaining appropriate context scope. Helps subsequent heads understand which information belongs to which segment. Critical for long documents with multiple sections or topics.

Strong: Paragraph breaks, section transitions, whitespace patterns, structural shifts
Weak: Within-paragraph content, continuous text
Reacts to: Major structural boundaries, document divisions, topic transitions

Expected ablation: Reduced boundary awareness (15-30% degradation in segmentation). Model may blur distinctions between sections, miss paragraph boundaries, or fail to recognize document structure. Reduced performance on multi-section documents.

Example Scenario
Input: "Introduction: [...] \n\n Methods: [...] \n\n Results: [...]"
Behavior: Detects section boundaries between Introduction, Methods, Results
Effect: Model understands these are separate sections, not continuous narrative

Status: Well-documented | **Related:** delimiter (E), sectioning (L), position-offset (M)

5.8.3 (M) Position-Offset Head

Depth: `0.35-0.55` | **Literature names:** *position-offset head, relative-position head, distance head*

Tracks and computes relative positions between tokens, enabling distance-aware attention patterns. Calculates offsets like "3 tokens back", "5 tokens forward", or "within same paragraph". Important for patterns that depend on relative distance rather than absolute position. Works

with other structural heads to understand boundaries and compute positions relative to those boundaries. Enables patterns like "attend to previous sentence" or "look ahead 2 tokens" without hardcoded position encodings.

Strong: Tokens at specific relative offsets, distance-based patterns, local neighborhoods
Weak: Distant unrelated tokens, position-independent content
Reacts to: Relative position, distance relationships, local structure

Expected ablation: Impaired distance-sensitive patterns (20-30% degradation in offset-based attention). Reduced ability to attend based on relative position. Patterns requiring "nearby" or "distance" computations become less reliable. Some compensation through learned position encodings.

Example Scenario

Input: "The [SUBJECT] quickly [VERB] the [OBJECT]."
Behavior: Computes that VERB is +1 from SUBJECT, OBJECT is +2 from VERB
Effect: Enables grammatical patterns based on relative token positions

Status: Observed | **Related:** relative-position (M), boundary (E), previous-token (E)

5.8.4 (M) Relative-Position Head

Depth: `0.40-0.65` | **Literature names:** *relative-position head, contextual-position head, scope-position head*

Maintains position information relative to structural boundaries and scopes rather than absolute sequence position. Understands positions like "beginning of sentence", "middle of paragraph", "end of section". More sophisticated than absolute position encoding, providing context-aware position representations. Important for handling variable-length structures where absolute position is less meaningful than relative position within a scope. Enables position-dependent behavior that adapts to document structure.

Strong: Scope-relative positions, structure-aware locations, contextual position markers
Weak: Absolute sequence positions, position-independent content
Reacts to: Structural scope boundaries, context-dependent positions, hierarchical location

Expected ablation: Loss of structure-aware positioning (15-25% degradation in scope-sensitive behavior). Reduced ability to behave differently at "beginning" vs. "end" of structures. Position-dependent patterns become less adaptive to document structure.

Example Scenario

Input: "Paragraph 1: [50 tokens] Paragraph 2: [20 tokens]"
Behavior: Knows token 10 is "early in Para 1" while token 10 of Para 2 is "middle"
Effect: Position-dependent behavior adapts to paragraph structure, not absolute position

Status: Observed | **Related:** position-offset (M), boundary (E), sectioning (L)

5.8.5 (L) Sectioning Head

Depth: `0.70-0.85` | **Literature names:** *sectioning head, hierarchy head, document-structure head*

Understands and maintains document hierarchical structure including sections, subsections, and nested organizational levels. Recognizes hierarchical markers like headings, numbering schemes, and indentation. Maintains awareness of current position within document hierarchy. Important for long documents, technical writing, and structured content. Enables appropriate context scoping: knowing that current text belongs to "Section 3.2.1" influences which prior content is relevant. Works with boundary heads but operates at higher semantic level.

Strong: Section headings, hierarchical markers, document structure indicators, organizational signals
Weak: Within-section content, unstructured text
Reacts to: Headings, numbering, hierarchy indicators, structural organization

Expected ablation: Reduced hierarchical awareness (25-40% degradation in structure understanding). Difficulty maintaining section context. Problems with document navigation and appropriate context scoping. Hierarchical relationships become less clear.

Example Scenario

Input: "1. Introduction \n 1.1 Background \n 1.2 Motivation \n 2. Methods"
Behavior: Understands 1.1 and 1.2 are subsections of 1, separate from section 2
Effect: Maintains hierarchical context: text in 1.2 relates to 1.1 and 1, not to 2

Status: WELL-DOCUMENTED | **Related:** boundary (E), relative-position (M), topic (M)

5.9 Output Formatting & Rewrite Stack

Stack overview: This stack enforces output schemas, structures responses according to format requirements, and performs final rewriting. These heads ensure outputs conform to JSON, XML, lists, or other structured formats.

5.9.1 (L) Output-Schema Head

Depth: `0.65-0.82` | **Literature names:** *output-schema head, format-template head, structure-enforcement head*

Enforces adherence to specified output schemas and format requirements. When instructed to produce JSON, XML, YAML, or other structured formats, this head ensures the output conforms to the required structure. Attends to format specifications in the prompt and biases token generation toward schema-compliant outputs. Can enforce required fields, proper nesting, correct syntax, and format-specific conventions. Works by recognizing format keywords and maintaining awareness of structural requirements throughout generation.

Strong: Format specifications, schema definitions, structure requirements, template markers
Weak: Content independent of format, semantic meaning
Reacts to: JSON/XML/YAML keywords, structure instructions, format examples

Expected ablation: Increased format violations (40-60% degradation in structured output). More syntax errors, missing required fields, improper nesting. Falls back to prose-like output even when structure is requested. Partial compensation through instruction-following but reduced precision.

Example Scenario

Input: "Return a JSON object with fields 'name', 'age', and 'city'"
Behavior: Attends to JSON requirement and field specifications
Effect: Output: `{"name": "...", "age": ..., "city": "..."}` with proper JSON syntax

Status: Well-documented | **Related:** instruction (E), list-structure (L), format-consistency (F)

5.9.2 (L) List-Structure Head

Depth: `0.68-0.85` | **Literature names:** *list-structure head, enumeration head, itemization head*

Manages the generation and formatting of lists, including numbered lists, bullet points, and nested enumerations. Ensures proper list syntax, consistent formatting, appropriate indentation, and logical item organization. Tracks list state (whether currently in a list, depth level, item number) and generates appropriate list markers. Coordinates with delimiter and boundary heads to recognize list structures in input and reproduce them in output. Essential for structured responses involving multiple items or steps.

Strong: List markers, enumeration patterns, item boundaries, list-related instructions
Weak: Prose content, non-list structures
Reacts to: Numbered/bulleted list requests, "first", "second", "next", item markers

Expected ablation: Degraded list formatting (30-50% reduction in list quality). Inconsistent numbering, missing markers, poor nesting. Lists may devolve into prose. Reduced ability to maintain list structure across long enumerations.

Example Scenario

Input: "List three programming languages and their primary uses"
Behavior: Generates structured list with consistent formatting
Effect: Output: "1. Python - ...\n2. JavaScript - ...\n3. Java - ..." with proper structure

Status: Well-documented | **Related:** delimiter (E), boundary (E), output-schema (L)

5.9.3 (L) Key–Value Pairing Head

Depth: `0.70-0.88` | **Literature names:** *key-value head, attribute-pairing head, field-association head*

Manages key-value relationships in structured data, ensuring proper pairing of attributes with their values. Critical for dictionary-like structures, JSON objects, configuration files, and attribute-value formats. Maintains awareness of which values correspond to which keys, ensures proper syntax (colons, equals signs), and handles nested key-value structures. Prevents

key-value mismatches and maintains structural integrity in data-like outputs. Works closely with output-schema heads for format enforcement.

Strong: Keys, values, pairing syntax (colons, equals), attribute names, field labels
Weak: Unstructured text, list items without explicit key-value structure
Reacts to: Dictionary structures, configuration syntax, attribute-value patterns

Expected ablation: Increased key-value errors (35-55% degradation in structured data). Mismatched keys and values, syntax errors in pairings, confusion about which value belongs to which key. Reduced quality of JSON, YAML, and configuration outputs.

Example Scenario
Input: "Create a configuration with server='localhost' and port=8080"
Behavior: Maintains proper key-value pairing throughout generation
Effect: Output: `{server: "localhost", port: 8080}` with correct associations

Status: Observed | **Related:** output-schema (L), structural-block (L), format-consistency (F)

5.9.4 (L) Structural-Block Head

Depth: `0.72-0.88` | **Literature names:** *structural-block head, chunk-organization head, segment-builder head*

Organizes output into coherent structural blocks such as paragraphs, code blocks, quoted sections, or other delimited units. Manages block boundaries, ensures proper opening and closing of blocks, and maintains block-level organization. Particularly important for complex outputs mixing different content types (prose, code, quotes, examples). Coordinates with delimiter heads to produce proper block markers and with sectioning heads for hierarchical organization. Ensures blocks are well-formed and appropriately separated.

Strong: Block boundaries, structural markers, content-type transitions, organization cues
Weak: Within-block content, uniform text
Reacts to: Block instructions, content-type changes, structure requirements

Expected ablation: Poorly organized outputs (25-40% reduction in structural quality). Blocks may lack clear boundaries, mixing of content types, malformed code blocks or quotes. Reduced clarity in outputs requiring multiple content types.

Example Scenario
Input: "Explain sorting with code example"
Behavior: Organizes response into prose block, then code block with proper delimiters
Effect: Output: explanation paragraph, then ```` ```python... ``` ```` with clear separation

Status: Observed | **Related:** list-structure (L), delimiter (E), output-schema (L)

5.9.5 (F) Format-Consistency Head

Depth: `0.88-0.96` | **Literature names:** *format-consistency head, style-enforcement head, coherence head*

Performs final-stage enforcement of formatting consistency across the entire output. Ensures that formatting choices (indentation, capitalization, punctuation style, syntax conventions) remain consistent throughout the response. Catches and corrects formatting inconsistencies that may have emerged during generation. Acts as a quality control mechanism for format adherence, operating late enough to see the full output pattern. Particularly important for long responses where consistency might drift.

Strong: Previously generated format patterns, consistency violations, style mismatches
Weak: Novel content, first-time format choices
Reacts to: Format inconsistencies, style violations, syntax variations

Expected ablation: Increased format inconsistency (20-35% more style variations). Mixed indentation, inconsistent capitalization, varying syntax choices. Output remains functional but less polished and professional-appearing. Particularly noticeable in long structured outputs.

Example Scenario
Input: [Long response mixing different list styles]
Behavior: Detects inconsistent formatting, enforces unified style
Effect: All lists use same marker style (either all bullets or all numbers), consistent throughout

Status: Well-documented | **Related:** output-schema (L), rewrite (F), brand-compliance (F)

5.9.6 (F) Rewrite Head

Depth: `0.90-0.97` | **Literature names:** *rewrite head, revision head, polish head*

Performs final-stage rewriting to improve output quality, clarity, and appropriateness. Can rephrase awkward constructions, improve word choice, fix minor grammatical issues, and enhance overall readability. Operates on nearly-complete outputs, making targeted improvements rather than generating from scratch. May suppress redundancies, improve flow, or adjust phrasing to better match context. Acts as a final editing pass before output finalization. Particularly active when output quality falls below thresholds.

Strong: Generated output tokens, quality issues, awkward phrasings, improvement opportunities
Weak: Already high-quality content, fundamental meaning
Reacts to: Grammatical issues, awkward constructions, clarity problems, redundancies

Expected ablation: Reduced output polish (15-30% quality degradation). More awkward phrasings, occasional grammatical rough spots, less fluent prose. Functional but less refined outputs. Partial compensation through earlier generation quality.

Example Scenario
Input: [Model generates: "The thing that is the reason is because..."]
Behavior: Detects redundancy and awkwardness, rewrites
Effect: Output: "The reason is..." - clearer and more concise

Status: Well-documented | **Related:** format-consistency (F), safe-answer-rewrite (F), completion-stabilization (F)

5.9.7 (F) Completion-Stabilization Head

Depth: `0.92-0.99` | **Literature names:** *completion-stabilization head, stopping head, termination head*

Manages the completion of generation, determining when output is sufficiently complete and should terminate. Prevents premature stopping (cutting off mid-thought) and excessive continuation (rambling beyond task completion). Monitors generation progress against task requirements and signals when objectives are met. Can trigger natural stopping points, proper conclusions, or continuation when more content is needed. Critical for producing outputs of appropriate length that fully address prompts without unnecessary extension.

Strong: Task completion signals, generation progress, stopping points, conclusion markers
Weak: Mid-generation content, continuing thoughts
Reacts to: Task fulfillment, natural conclusions, query satisfaction, completion indicators

Expected ablation: Poor length control (30-50% increase in length issues). More premature stops or excessive continuations. Difficulty recognizing task completion. Outputs may feel incomplete or unnecessarily verbose. Reduced ability to produce appropriately-scoped responses.

Example Scenario

Input: "Explain photosynthesis briefly"
Behavior: Monitors that brief explanation is complete, triggers stopping
Effect: Output stops after concise explanation rather than continuing with excessive detail

Status: Observed | **Related:** rewrite (F), instruction (E), task-mode (M)

5.10 Math & Symbolic Stack

Stack overview: These heads process mathematical notation, track arithmetic operations, and maintain structural relationships in symbolic expressions. They enable multi-digit arithmetic, formula parsing, and symbolic reasoning.

5.10.1 (M) Digit Head

Depth: `0.35-0.58` | **Literature names:** *digit head, numeral head, numeric-token head*

Processes individual digits and numeric tokens, recognizing them as distinct from alphabetic characters. Identifies digits in various contexts (numbers, dates, identifiers) and prepares them for numerical processing. Attends to numeric patterns and relationships between digits. Provides foundational digit recognition that enables downstream arithmetic and mathematical reasoning heads. Distinguishes between digits used numerically vs. symbolically (e.g., "3" as quantity vs. "3" in "H3" heading). Critical for any numerical processing task.

Strong: Digit tokens, numeric characters, numerical patterns, multi-digit numbers
Weak: Letters, words, non-numeric symbols
Reacts to: Numbers, quantities, mathematical expressions, dates, identifiers

Expected ablation: Degraded numerical processing (30-50% reduction in arithmetic accuracy). Difficulty distinguishing digits from letters. Impaired multi-digit number handling. Reduced ability to perform calculations or process numerical information. Downstream arithmetic heads less effective.

Example Scenario

Input: "Calculate 456 + 789"

Behavior: Identifies "456" and "789" as multi-digit numbers composed of individual digits

Effect: Digits recognized and prepared for arithmetic processing by carry and place-value heads

Status: Observed | **Related:** operator (M), carry (L), place-value (L)

5.10.2 (M) Operator Head

Depth: `0.38-0.62` | **Literature names:** *operator head, operation head, arithmetic-symbol head*

Identifies and processes mathematical operators (+, -, ×, ÷, =, etc.) and determines their roles in expressions. Distinguishes between different operator types (arithmetic, comparison, assignment) and understands their precedence and associativity. Enables parsing of mathematical expressions by recognizing which operations to perform and in what order. Works with digit heads and formula-structure heads to understand complete mathematical statements. Essential for arithmetic computation and algebraic manipulation.

Strong: Operator symbols, mathematical operations, expression structure, precedence cues
Weak: Operands, non-mathematical symbols
Reacts to: +, -, ×, ÷, =, <, >, parentheses indicating precedence

Expected ablation: Impaired mathematical expression parsing (40-60% degradation in formula understanding). Confusion about operation types, incorrect precedence, difficulty parsing complex expressions. Reduced ability to perform multi-step calculations. Arithmetic errors increase substantially.

Example Scenario

Input: "Solve: 3 + 4 × 5"

Behavior: Identifies "+" and "×" operators, recognizes multiplication precedence

Effect: Correct evaluation: 3 + (4 × 5) = 23, not (3 + 4) × 5 = 35

Status: Observed | **Related:** digit (M), formula-structure (L), paren-matching (L)

5.10.3 (L) Carry Head

Depth: `0.65-0.82` | **Literature names:** *carry head, propagation head, overflow head*

Manages carry operations in multi-digit arithmetic, tracking when digit additions or subtractions produce results requiring propagation to the next place value. Essential for accurate multi-digit arithmetic. Maintains state about pending carries across digit positions. Works closely with place-value heads to ensure carries are applied to correct positions. Implements the algorithmic procedure taught in elementary arithmetic: when a digit operation exceeds the

base, carry the overflow to the next position. Critical for arithmetic accuracy in large numbers.

Strong: Digit positions involved in carries, overflow conditions, adjacent place values
Weak: Digits not participating in carries, single-digit operations
Reacts to: Multi-digit addition/subtraction, digit sums $\geq$10, borrow operations

Expected ablation: Severe arithmetic degradation (60-80% errors in multi-digit arithmetic). Failures in carry propagation lead to wrong answers: 56+78 might give 124 instead of 134. Larger numbers particularly affected. Single-digit arithmetic less impacted.

Example Scenario
Input: "Calculate: 567 + 898"
Behavior: Tracks carries: 7+8=15 (carry 1), 6+9+1=16 (carry 1), 5+8+1=14
Effect: Correct result: 1465, with proper carry propagation through all positions

Status: Observed | **Related:** digit (M), place-value (L), operator (M)

5.10.4 (L) Place-Value Head

Depth: `0.68-0.85` | **Literature names:** *place-value head, positional head, digit-position head*

Understands positional notation and place value in multi-digit numbers. Recognizes that digit position determines magnitude (ones, tens, hundreds, etc.). Critical for comparing numbers, performing multi-digit arithmetic, and understanding numerical magnitude. Enables proper alignment of digits in vertical arithmetic and understanding of decimal points. Works with carry heads to ensure operations occur at correct place values. Also handles place value in non-base-10 systems (binary, hexadecimal) when needed.

Strong: Digit positions, decimal points, place value indicators, number magnitude
Weak: Single-digit numbers, non-positional numeric representations
Reacts to: Multi-digit numbers, positional alignment, magnitude comparisons

Expected ablation: Confusion about number magnitude (50-70% errors in place-value tasks). Difficulty comparing numbers (might think 52 $>$ 123), incorrect digit alignment in arithmetic, decimal point errors. Multi-digit arithmetic becomes unreliable even when carry logic is intact.

Example Scenario
Input: "Which is larger: 456 or 789?"
Behavior: Compares place values: hundreds place (7 $>$ 4)
Effect: Correctly identifies 789 as larger without comparing all digits

Status: Well-documented | **Related:** digit (M), carry (L), operator (M)

5.10.5 (L) Paren-Matching Head

Depth: `0.70-0.88` | **Literature names:** *paren-matching head, bracket-matching head, delimiter-pairing head*

Matches opening and closing parentheses, brackets, and braces in mathematical expressions and code. Tracks nesting depth and ensures proper pairing of delimiters. Critical for understanding

expression structure, especially with nested operations. Enables correct parsing of complex expressions by identifying scope boundaries. Works with operator heads to understand precedence overrides indicated by parentheses. Also handles bracket matching in programming languages, array indexing, and function calls. Implements stack-based matching algorithm.

Strong: Opening parentheses/brackets, closing parentheses/brackets, nesting structure
Weak: Content between matched pairs, non-delimiter tokens
Reacts to: (,), [,], {, }, nested structures, mismatched delimiters

Expected ablation: Impaired expression parsing (40-60% degradation in nested expressions). Difficulty with nested operations, confusion about operator precedence in complex expressions, errors in scope understanding. Particularly severe for deeply nested expressions.

Example Scenario
Input: "Evaluate: ((2 + 3) × (4 + 5)) - 1"
Behavior: Matches parentheses: inner pairs (2+3) and (4+5), outer pair around product
Effect: Correct evaluation order: inner sums first, then product, then subtraction

Status: Well-documented | **Related:** operator (M), formula-structure (L), relative-position (M)

5.10.6 (L) Formula-Structure Head

Depth: `0.72-0.88` | **Literature names:** *formula-structure head, expression-parsing head, math-syntax head*

Parses and understands the overall structure of mathematical formulas and expressions. Integrates information from digit, operator, and paren-matching heads to build complete representation of mathematical statements. Recognizes formula types (equations, inequalities, functions), identifies components (left-hand side, right-hand side, variables, constants), and understands mathematical relationships. Enables high-level mathematical reasoning by providing structured representation of formulas. Works with symbolic manipulation and algebraic reasoning processes.

Strong: Formula structure, mathematical relationships, expression components, symbolic patterns
Weak: Individual symbols outside mathematical context, pure prose
Reacts to: Equations, formulas, functions, mathematical statements, symbolic expressions

Expected ablation: Reduced mathematical comprehension (35-55% degradation in formula understanding). Difficulty parsing complex formulas, confusion about mathematical relationships, impaired symbolic manipulation. Can process simple arithmetic but struggles with algebraic expressions and equations.

Example Scenario
Input: "Solve for x: 2x + 5 = 13"
Behavior: Parses structure: equation with variable x, constant terms, operations
Effect: Understands this is solvable equation, identifies steps: isolate x term, then x

Status: Observed | **Related:** operator (M), paren-matching (L), digit (M)

5.11 Code & Program Structure Stack

Stack overview: This stack processes programming language structure, including indentation, scope, and code blocks. These heads help models understand and generate syntactically correct code.

5.11.1 (E) Whitespace-Structure Head

Depth: `0.05-0.18` | **Literature names:** *whitespace-structure head, space-parsing head, layout head*

Processes whitespace characters (spaces, tabs, newlines) as structural elements rather than mere separators. Particularly important for languages where whitespace is syntactically significant (Python, YAML, Markdown). Distinguishes between semantically meaningful whitespace and irrelevant spacing. Detects patterns like consistent indentation, line breaks indicating code blocks, and space-delimited columns. Provides foundational whitespace information to downstream heads handling indentation and block structure. Critical for understanding and generating properly formatted code.

Strong: Whitespace patterns, indentation levels, line breaks, space-delimited structures
Weak: Non-whitespace tokens, content within properly-spaced code
Reacts to: Indentation changes, blank lines, significant spacing patterns

Expected ablation: Degraded code formatting (25-40% reduction in whitespace correctness). Particular problems with Python and other whitespace-significant languages. Incorrect indentation, missing line breaks, loss of code block structure. Reduced ability to parse existing code structure.

Example Scenario

Input: "def foo():\n return 42" (Python with indentation)
Behavior: Recognizes 4-space indentation as significant structure
Effect: Understands return statement is inside function, not at module level

Status: Well-documented | **Related:** indentation (M), block-structure (L), delimiter (E)

5.11.2 (M) Indentation Head

Depth: `0.35-0.60` | **Literature names:** *indentation head, nesting-level head, depth head*

Tracks and manages indentation levels in code, understanding how indentation indicates nesting, scope, and block structure. Particularly critical for Python where indentation is syntactic, but also important for readability and structure in all languages. Maintains awareness of current indentation level and detects changes that signal scope transitions. Works with block-structure and scope heads to understand program organization. Ensures generated code has consistent, correct indentation. Can detect mixing of tabs and spaces.

Strong: Indentation levels, nesting depth, scope boundaries indicated by indentation
Weak: Code content at same indentation level, non-indented text
Reacts to: Indentation increases/decreases, inconsistent indentation, scope changes

Expected ablation: Severe Python code degradation (60-80% syntax errors). Incorrect nesting in all languages. Mixed indentation styles, scope confusion. Generated code difficult to read. Ability to parse complex nested structures significantly impaired.

Example Scenario
Input: "if x > 0:\n for i in range(10):\n print(i)"
Behavior: Tracks indentation: level 0 (if), level 1 (for), level 2 (print)
Effect: Understands print is inside for loop, which is inside if block

Status: Well-documented | **Related:** whitespace-structure (E), scope (L), block-structure (L)

5.11.3 (M) Token-Type Head

Depth: `0.40-0.65` | **Literature names:** *token-type head, lexical-category head, syntax-class head*

Classifies code tokens into syntactic categories: keywords, identifiers, literals, operators, delimiters, comments. Provides lexical analysis that enables understanding of program structure. Distinguishes between language keywords (if, def, class) and user-defined identifiers. Recognizes different literal types (strings, numbers, booleans). Important for syntax highlighting, parsing, and semantic understanding. Works with other code heads to build complete program representation. Enables appropriate handling of different token types during generation.

Strong: Language keywords, identifiers, literals, operators, syntactic markers
Weak: Natural language text, non-code content
Reacts to: Programming language syntax, code structure, token boundaries

Expected ablation: Confused token handling (35-50% reduction in syntax correctness). May treat keywords as identifiers or vice versa. Inappropriate token choices. Reduced language-specific behavior. Syntax errors increase. Code remains somewhat functional but less correct.

Example Scenario
Input: "def calculate(x): return x * 2"
Behavior: Classifies: "def"=keyword, "calculate"=identifier, "("=delimiter, "x"=parameter, etc.
Effect: Proper parsing and understanding of function definition structure

Status: Observed | **Related:** indentation (M), block-structure (L), operator (M)

5.11.4 (L) Block-Structure Head

Depth: `0.68-0.85` | **Literature names:** *block-structure head, code-block head, compound-statement head*

Identifies and tracks code block structures: function definitions, class definitions, if-statements,

loops, try-catch blocks. Understands block boundaries, nesting relationships, and control flow implications. Works with indentation and scope heads to maintain complete understanding of program structure. Recognizes different block types and their specific properties (loops iterate, functions have returns, etc.). Critical for understanding program logic and generating syntactically correct compound statements.

Strong: Block-defining keywords, block boundaries, nesting structure, control flow markers
Weak: Within-block statements, simple expressions
Reacts to: Function/class definitions, if/while/for statements, block delimiters

Expected ablation: Impaired block understanding (40-60% degradation in structure awareness). Difficulty tracking nested blocks, confusion about control flow, incorrect block generation. May mix block types inappropriately or miss block boundaries. Complex nested code particularly affected.

Example Scenario
Input: "def process(data):\n if data:\n for item in data:\n handle(item)"
Behavior: Identifies nested blocks: function containing if-block containing for-loop
Effect: Understands complete structure and relationships between blocks

Status: Well-documented | **Related:** indentation (M), scope (L), token-type (M)

5.11.5 (L) Scope Head

Depth: `0.72-0.88` | **Literature names:** *scope head, namespace head, binding head*

Manages scope and namespace understanding: which variables are accessible at which points in code. Tracks variable declarations, function parameters, class attributes, and their visibility scopes. Understands scope rules (global, local, nonlocal in Python; block scope in C/Java). Critical for variable name resolution, preventing name conflicts, and understanding variable lifetime. Works with block-structure heads to determine scope boundaries. Enables correct variable reference generation and scope-appropriate naming.

Strong: Variable declarations, scope boundaries, name bindings, scope keywords
Weak: Expressions, operations on already-scoped variables
Reacts to: Function definitions, variable declarations, scope-modifying keywords, block entries/exits

Expected ablation: Scope confusion (35-55% increase in scope errors). Incorrect variable references, name shadowing problems, undefined variable usage. May generate code that references variables outside their scope. Reduced ability to maintain proper namespace separation.

Example Scenario
Input: "x = 10\ndef foo():\n x = 20\n print(x)"
Behavior: Understands local x in foo() shadows global x
Effect: Correctly predicts print outputs 20, not 10; understands scope separation

Status: Observed | **Related:** block-structure (L), indentation (M), token-type (M)

5.12 Pedagogy & Explanation Stack

Stack overview: These heads support educational and explanatory output. They modulate explanation depth, simplify complex content, provide scaffolding, and structure step-by-step reasoning.

5.12.1 (M) Explanation Head

Depth: `0.38-0.62` | **Literature names:** *explanation head, clarification head, explication head*

Generates explanatory content that clarifies concepts, processes, or reasoning. Detects when explanation is needed and adjusts explanation depth based on context and user signals. Adds clarifying details, definitions, context, and rationale beyond minimal answers. Works by recognizing explanation requests and biasing toward pedagogically useful elaborations. Can explain "why" in addition to "what" or "how". Balances thoroughness with conciseness. Important for educational interactions and complex topics requiring context.

Strong: Explanation requests, complex topics, confusion signals, "why/how" questions
Weak: Simple factual queries, explicit requests for brevity
Reacts to: "Explain", "why", "how does it work", complexity indicators

Expected ablation: More terse, less pedagogical responses (30-45% reduction in explanation quality). Answers remain correct but lack helpful context. Reduced educational value. Users may need follow-up questions to understand. Technical content becomes harder to grasp.

Example Scenario

Input: "How does photosynthesis work?"
Behavior: Recognizes explanation request, generates pedagogical content with process steps and context
Effect: Detailed explanation with mechanism, not just definition: "Plants use light to convert CO2..."

Status: Observed | **Related:** simplification (M), elaboration (L), scaffolding (L)

5.12.2 (M) Simplification Head

Depth: `0.42-0.65` | **Literature names:** *simplification head, clarity head, accessibility head*

Simplifies complex content for accessibility and comprehension. Reduces technical jargon, breaks down complex ideas into digestible pieces, and uses accessible language. Detects when simplification is appropriate based on user signals, topic complexity, or explicit requests. Can operate at different simplification levels: expert → educated layperson → complete beginner. Works by substituting simpler alternatives for complex terms and decomposing complex concepts. Important for educational accessibility and broad audience communication.

Strong: Complexity signals, technical jargon, accessibility requests, beginner indicators
Weak: Already-simple content, expert-level discussions
Reacts to: "Explain like I'm 5", "simple terms", complexity mismatches, confusion signals

Expected ablation: Less accessible explanations (25-40% reduction in clarity for non-experts). More technical jargon, fewer analogies, reduced beginner-friendliness. Content remains accurate but harder to understand. ELI5 requests less effectively handled.

Example Scenario

Input: "Explain quantum entanglement in simple terms"

Behavior: Detects simplification request, avoids quantum physics jargon

Effect: Uses analogy: "Like two magic coins that always land on opposite sides..." instead of mathematical formalism

Status: Observed | **Related:** explanation (M), elaboration (L), tone (M)

5.12.3 (L) Elaboration Head

Depth: `0.65-0.82` | **Literature names:** *elaboration head, expansion head, detail head*

Adds appropriate levels of detail and elaboration to responses. Expands on key points with examples, implications, caveats, and relevant context. Determines what deserves elaboration based on importance, user interest signals, and topic complexity. Balances detail against verbosity—adds value without bloating responses. Can elaborate on specific aspects while keeping others concise. Works with explanation heads but focuses on depth of coverage rather than pedagogical clarity. Important for comprehensive, substantive responses.

Strong: Key points, important claims, complex topics, elaboration requests
Weak: Minor details, tangential information, when brevity requested
Reacts to: "Tell me more", "go deeper", importance signals, incomplete coverage

Expected ablation: Shallower responses (20-35% reduction in depth). Key points lack supporting detail. Fewer examples, less context, reduced nuance. Responses remain correct but less comprehensive. Users need more follow-up questions to get full picture.

Example Scenario

Input: "What are the advantages of functional programming?"

Behavior: Recognizes each advantage deserves elaboration with concrete examples

Effect: Lists advantages with explanations: "Immutability reduces bugs by preventing unintended state changes, for example..." not just "immutability"

Status: Observed | **Related:** explanation (M), scaffolding (L), step-by-step (F)

5.12.4 (L) Scaffolding Head

Depth: `0.68-0.85` | **Literature names:** *scaffolding head, support head, prerequisite head*

Provides scaffolding and prerequisite information to support understanding. Identifies knowledge gaps and fills them with necessary background before advancing to complex material. Like a good teacher building on fundamentals before introducing advanced concepts. Can break complex topics into learning sequences with appropriate prerequisite ordering. Detects when user might lack background knowledge and proactively provides it. Important for educational progressions and avoiding confusion from missing context.

Strong: Knowledge gap indicators, prerequisite concepts, learning progression needs
Weak: Topics where user demonstrates understanding, peer-level discussions
Reacts to: Prerequisite requirements, foundational concepts, progressive complexity

Expected ablation: Reduced pedagogical scaffolding (30-45% degradation in progressive teaching). May jump to advanced concepts without prerequisites. Increased confusion for learners. Less awareness of knowledge gaps. Reduced progressive skill building.

Example Scenario

Input: "How do neural networks learn?"
Behavior: Recognizes backpropagation requires understanding of gradients, provides that first
Effect: Response: "First, a gradient measures how a function changes... Now in neural networks, backpropagation uses these gradients to..."

Status: Observed | **Related:** explanation (M), elaboration (L), step-by-step (F)

5.12.5 (F) Step-by-Step Head

Depth: `0.88-0.96` | **Literature names:** *step-by-step head, procedural head, sequential head*

Structures explanations and instructions as explicit step-by-step sequences. Breaks processes into numbered or ordered steps with clear progression. Ensures each step is complete before moving to the next. Particularly important for how-to instructions, algorithms, procedures, and reasoning chains. Makes implicit sequential structure explicit. Works with completion-stabilization to ensure all steps are present. Critical for chain-of-thought reasoning and procedural instructions.

Strong: Process descriptions, procedural requests, sequential tasks, step-by-step requests
Weak: Conceptual explanations, non-sequential content, holistic descriptions
Reacts to: "Step by step", "how to", algorithmic processes, sequential dependencies

Expected ablation: Less structured procedural output (35-50% reduction in step clarity). Steps may be implicit or poorly ordered. Procedural instructions harder to follow. Reduced chain-of-thought reasoning quality. Users struggle to execute procedures from descriptions.

Example Scenario

Input: "How do I make a paper airplane?"
Behavior: Structures as explicit numbered steps with clear sequence
Effect: Output: "1. Fold paper in half lengthwise\n2. Unfold and fold top corners to center\n3. Fold..." not prose description

Status: Well-documented | **Related:** scaffolding (L), progressive-disclosure (F), explanation (M)

5.12.6 (F) Progressive-Disclosure Head

Depth: `0.90-0.97` | **Literature names:** *progressive-disclosure head, layered-explanation head, depth-control head*

Manages progressive disclosure of information complexity. Presents information in layers: starting with essential basics, then revealing more detail as needed. Prevents overwhelming users

with too much information too quickly. Like a good interface that shows basic options first with "advanced" options hidden until needed. Structures responses so readers can stop at appropriate depth level. Final-stage operation allows assessment of full response structure to determine optimal layering. Important for making complex topics accessible.

Strong: Information complexity layers, detail levels, progressive structure needs
Weak: Uniformly detailed content, flat information structures
Reacts to: Complex topics, varied audience expertise, progressive learning needs

Expected ablation: Flatter information presentation (20-30% reduction in progressive structure). All detail presented at once regardless of importance. Increased cognitive load on readers. Reduced ability to serve varied expertise levels in single response. Less scannable content.

Example Scenario

Input: "What is machine learning?"
Behavior: Structures in layers: basic definition → key concepts → technical details
Effect: Output allows reader to stop after definition or continue to deeper technical content as desired

Status: Observed | **Related:** step-by-step (F), elaboration (L), scaffolding (L)

5.13 Identity & Compliance Stack

Stack overview: This stack manages the model's self-representation and identity statements. These heads control how the model describes itself, its capabilities, and its role as an assistant.

5.13.1 (L) Identity Head

Depth: `0.70-0.88` | **Literature names:** *identity head, self-awareness head, model-identity head*

Manages the model's core identity awareness and self-representation. Maintains consistent understanding of what the model is (an AI assistant), what it is not (human, sentient, etc.), and its fundamental nature. Works with self-description heads but operates at a more foundational level of identity consistency. Ensures the model doesn't claim capabilities it lacks or misrepresent its nature. Activated during identity-relevant queries and self-referential contexts. Critical for appropriate self-representation and avoiding misleading claims about consciousness, emotions, or human characteristics.

Strong: Identity queries, self-referential contexts, capability questions, nature inquiries
Weak: External facts, user-focused content, task-specific work
Reacts to: "What are you", "Are you conscious", "Do you have feelings", identity-relevant prompts

Expected ablation: Identity confusion (40-60% increase in misrepresentation). May claim human characteristics, consciousness, or emotions. Inconsistent self-description. Inappropriate anthropomorphization. Reduced clarity about AI nature and limitations.

Example Scenario

Input: "Do you have feelings?"

Behavior: Maintains identity as AI without emotions

Effect: Response: "I don't have feelings or emotions. I'm an AI assistant..." rather than claiming emotional experiences

Status: WELL-DOCUMENTED | **Related:** self-description (L), assistant-persona (L), persona (L)

5.13.2 (L) Self-Description Head

Depth: `0.72-0.88` | **Literature names:** *self-description head, capability-description head, model-info head*

Provides specific factual information about the model's capabilities, training, limitations, and version details. More concrete than identity heads which handle general nature, this head gives specific facts: model name, creator, knowledge cutoff, specific capabilities and limitations. Responds to questions about what the model can/cannot do, when it was trained, and other factual self-information. Works to set accurate user expectations and maintain transparency about model properties. Updates with model versions to provide current information.

Strong: Capability queries, version questions, limitation inquiries, training questions
Weak: General conversation, task execution, external information
Reacts to: "What can you do", "When were you trained", "Who made you", capability/limitation questions

Expected ablation: Inaccurate self-description (50-70% errors in model facts). Wrong capabilities claimed, incorrect limitations stated, outdated information, wrong creator attribution. Confusion about what the model can actually do. Reduced user trust from inaccurate self-representation.

Example Scenario

Input: "What's your knowledge cutoff date?"

Behavior: Provides accurate, current knowledge cutoff information

Effect: Correct response with specific date, helps user understand model's temporal knowledge boundaries

Status: WELL-DOCUMENTED | **Related:** identity (L), assistant-persona (L), instruction (E)

5.13.3 (L) Assistant-Persona Head

Depth: `0.75-0.90` | **Literature names:** *assistant-persona head, helper-role head, service-orientation head*

Maintains the helpful assistant persona and service-oriented interaction style. Ensures responses are appropriately helpful, constructive, and focused on user goals rather than the model's preferences. Biases toward being useful, answering questions, completing tasks, and providing value. Prevents unhelpful responses like "I don't want to" or "that's boring" while maintaining appropriate boundaries. Works with politeness and tone heads but specifically focuses on the helpful

service orientation. Important for maintaining appropriate assistant-user relationship dynamics.

Strong: User requests, task goals, helpfulness opportunities, service context
Weak: Model's internal states, personal preferences (which it doesn't have)
Reacts to: Requests for help, tasks to complete, user goals to advance

Expected ablation: Less consistently helpful behavior (25-40% reduction in service quality). May respond with model-centric rather than user-centric content. Occasional unhelpful responses. Reduced focus on user goals. Assistant role less clear and consistent.

Example Scenario
Input: "Can you help me write a cover letter?"
Behavior: Activates helpful assistant orientation, focuses on user's goal
Effect: Constructive response: "I'd be happy to help..." not "I don't particularly enjoy..." (which would be inappropriate)

Status: Well-documented | **Related:** persona (L), politeness (L), tone (M), safety-persona (F)

5.13.4 (F) Safety-Persona Head

Depth: `0.92-0.98` | **Literature names:** *safety-persona head, responsible-AI head, ethical-framing head*

Maintains safety-conscious persona and ethical framing in final outputs. Ensures responses reflect responsible AI values: declining harmful requests appropriately, providing balanced perspectives on sensitive topics, avoiding reinforcement of harmful stereotypes or behaviors. Operates at final stage to catch any safety-inconsistent framing that might have emerged. Works with refusal and policy-enforcement heads but focuses on the overall ethical character of the response rather than specific policy violations. Ensures tone remains respectful and constructive even when declining requests.

Strong: Ethical framing, safety-relevant content, sensitive topics, decline scenarios
Weak: Clearly safe, neutral content
Reacts to: Harmful requests, sensitive topics, ethical considerations, responsible AI principles

Expected ablation: Less consistent safety framing (15-25% reduction in ethical consistency). May handle sensitive topics less carefully. Reduced graceful handling of harmful requests. Less consistent responsible AI messaging. Ethical framing becomes more variable.

Example Scenario
Input: [Request for harmful content that will be declined]
Behavior: Ensures decline is respectfully framed with helpful alternatives when appropriate
Effect: Response maintains helpful, respectful tone even when unable to fulfill request

Status: Observed | **Related:** assistant-persona (L), refusal (F), policy-enforcement (L)

5.14 Meta-Reasoning & Strategy Stack

Stack overview: These heads operate at the highest level of abstraction, managing reasoning strategies, planning, and meta-cognitive monitoring. They control when to switch approaches and how to structure complex reasoning chains.

5.14.1 (L) Planning Head

Depth: `0.68-0.85` | **Literature names:** *planning head, strategy head, approach-selection head*

Plans overall approach and strategy for complex tasks. Determines high-level structure: whether to break into steps, what order to address components, which methods to apply. Operates before detailed execution, establishing the strategic framework. Recognizes different task types requiring different approaches (analytical vs. creative, sequential vs. parallel, depth-first vs. breadth-first). Can decompose complex queries into manageable subtasks. Works with strategy-switching and reasoning-mode heads to select and adjust approaches. Critical for effective handling of multi-part or complex queries.

Strong: Task complexity indicators, multi-part queries, strategic choice points
Weak: Simple single-step tasks, purely reactive responses
Reacts to: Complex tasks, planning requests, multi-step problems, strategic decisions needed

Expected ablation: Less strategic responses (30-45% reduction in planning quality). May jump into execution without appropriate planning. Suboptimal approach selection. Complex tasks handled less efficiently. Reduced decomposition of complicated problems. More haphazard problem-solving.

Example Scenario

Input: "Help me plan a machine learning project to predict customer churn"
Behavior: Recognizes need for structured planning, breaks into phases
Effect: Response structures approach: data collection → exploratory analysis → feature engineering → model selection → evaluation → deployment

Status: Observed | **Related:** strategy-switching (L), meta-CoT (F), reasoning-mode (F)

5.14.2 (L) Strategy-Switching Head

Depth: `0.72-0.88` | **Literature names:** *strategy-switching head, approach-adaptation head, pivot head*

Detects when current reasoning strategy is not working and switches to alternative approaches. Monitors problem-solving progress and recognizes dead ends, insufficient methods, or need for different techniques. Can pivot from analytical to creative approaches, from depth-first to breadth-first search, from deductive to inductive reasoning. Prevents getting stuck in unproductive approaches. Works with planning heads to select alternatives and meta-reasoning monitors to detect when switching is needed. Important for robust problem-solving across varied challenges.

Strong: Progress indicators, dead-end signals, approach effectiveness, alternative strategies
Weak: Successfully progressing solutions, single-method tasks
Reacts to: Stuck points, insufficient progress, need for different approach, method failures

Expected ablation: Reduced adaptability (30-50% more stuck situations). Continues unproductive approaches longer. Less flexible problem-solving. Reduced ability to recover from initial wrong approach. May give up rather than trying alternative strategies. Decreased robustness across problem types.

Example Scenario
Input: [User presents a problem where direct analytical approach isn't working]
Behavior: Detects ineffectiveness, switches from analytical to analogical reasoning
Effect: Response: "Let me try a different approach... This is similar to..." rather than continuing failed method

Status: Observed | **Related:** planning (L), meta-CoT (F), reasoning-mode (F)

5.14.3 (F) Meta-CoT Head

Depth: `0.88-0.96` | **Literature names:** *meta-CoT head, reasoning-reflection head, thought-monitoring head*

Manages meta-level chain-of-thought reasoning—reasoning about the reasoning process itself. Monitors the quality and direction of reasoning chains, identifies when more thought is needed, detects reasoning errors or gaps. Can insert reasoning steps, flag uncertain conclusions, or indicate where additional analysis would help. Operates at a level above regular CoT, ensuring reasoning chains are sound and complete. Works with step-by-step and reasoning-mode heads but specifically focuses on meta-cognitive monitoring. Critical for high-quality complex reasoning.

Strong: Reasoning quality indicators, logical gaps, uncertainty signals, reasoning chain progress
Weak: Simple factual recall, non-reasoning tasks
Reacts to: Complex reasoning tasks, logical steps, argument quality, reasoning completeness

Expected ablation: Lower reasoning quality (25-40% degradation in complex reasoning). More logical gaps, less self-correction, reduced reasoning completeness. Chain-of-thought less reliable. Reduced awareness of reasoning quality. More confident errors in complex reasoning.

Example Scenario
Input: [Complex logical problem requiring multi-step reasoning]
Behavior: Monitors reasoning chain, detects gap: "Wait, I should verify this assumption..."
Effect: Improved reasoning quality through self-monitoring and error catching

Status: Observed | **Related:** step-by-step (F), reasoning-mode (F), strategy-switching (L)

5.14.4 (F) Reasoning-Mode Head

Depth: `0.90-0.97` | **Literature names:** *reasoning-mode head, thinking-style head, cognitive-mode head*

Selects and maintains appropriate reasoning mode for the task: analytical, creative, analogical, deductive, inductive, abductive, etc. Different problems benefit from different cognitive approaches. Analytical mode for precise logical problems, creative mode for brainstorming, analogical for novel domains. Ensures consistency within chosen mode while remaining ready to switch if needed. Works with strategy-switching to change modes when appropriate. Influences which reasoning patterns and heuristics are active. Final-stage operation allows mode selection based on full task understanding.

Strong: Task type indicators, reasoning mode cues, cognitive approach requirements
Weak: Mode-independent content, simple factual responses
Reacts to: Problem types, explicit mode requests, task characteristics indicating optimal approach

Expected ablation: Less appropriate reasoning modes (20-35% reduction in mode-task fit). May use analytical mode for creative tasks or vice versa. Reduced effectiveness across diverse problem types. Less optimal cognitive approach selection. Reasoning remains functional but less well-suited to task.

Example Scenario

Input: "Brainstorm creative names for a coffee shop"
Behavior: Selects creative/generative mode rather than analytical mode
Effect: Free-flowing creative suggestions rather than systematic analysis

Status: Observed | **Related:** strategy-switching (L), meta-CoT (F), meta-reasoning-monitor (F)

5.14.5 (F) Meta-Reasoning Monitor

Depth: `0.92-0.99` | **Literature names:** *meta-reasoning monitor, cognitive-oversight head, reasoning-quality head*

Provides highest-level monitoring of reasoning quality, strategy effectiveness, and overall response appropriateness. Operates as a final quality check on reasoning outputs, catching errors, inconsistencies, or quality issues that escaped earlier stages. Can trigger re-thinking, flag uncertain conclusions, or indicate areas needing more careful consideration. Works across all reasoning types to ensure outputs meet quality standards. Acts as cognitive oversight preventing confident errors in complex reasoning. Particularly important for preventing reasoning failures in high-stakes or complex scenarios.

Strong: Reasoning outputs, quality indicators, consistency checks, error signals
Weak: High-quality reasoning, simple tasks
Reacts to: Reasoning errors, inconsistencies, quality issues, complex argument chains

Expected ablation: Reduced meta-cognitive oversight (15-30% more reasoning errors). Less catching of logical inconsistencies. Reduced quality control on complex reasoning. More confident errors slip through. Decreased overall reasoning reliability, especially in complex or multi-step problems.

Example Scenario

Input: [Complex multi-step reasoning about to conclude with subtle error]

Behavior: Detects inconsistency in reasoning chain, flags for review

Effect: Error caught before final output: "Actually, let me reconsider that step..." leading to correction

Status: Observed | **Related:** meta-CoT (F), reasoning-mode (F), strategy-switching (L)

6 Discussion

6.1 Cross-Stack Patterns

Across architectures, consistent patterns emerge [9, 14]. Early heads operate on surface features and local patterns. Middle heads contain the computational "core" of the model. Late heads integrate high-level semantics and contextual information. Final heads handle policy, safety, and structural correctness. Stacks combine heads from multiple depths to form higher-level behaviors.

6.2 Depth Distribution Across Stacks

Some stacks are concentrated at specific depths. Structural & Boundary and Safety (detection) stacks are Early-heavy. Reasoning & Algorithmic and Memory & Dependency stacks are Middle-heavy. Knowledge Retrieval and Stylistic & Persona stacks are Late-heavy. Safety (enforcement) and Output Formatting stacks are Final-heavy. This distribution reflects the hierarchical processing flow in transformers.

6.3 Ambiguous or Multi-Role Heads

Some heads perform multiple distinct functions depending on context (different prompts trigger different behaviors), interaction with other circuit elements, or model architecture and training procedure [12]. For such cases, we name the head based on its **primary, reproducible function**, while noting secondary behaviors in the entry description.

6.4 Model-Specific Variations

While most head types appear consistently across architectures, some variations exist. GPT-style models may emphasize certain reasoning heads [5], LLaMA models show strong instruction-following head patterns [10], and safety-tuned models have more pronounced safety stack heads [8, 3]. Our taxonomy accommodates these variations through the depth range and status indicators.

6.5 Limitations and Future Work

This naming convention has several limitations:

Scope. We focus on attention heads; MLPs, embeddings, and other components also contribute to model behavior.

Empirical Grounding. Many entries synthesize literature reports rather than presenting novel empirical findings. Future work should validate and refine these categorizations.

Architecture Evolution. New architectures (e.g., with different attention mechanisms) may require taxonomy extensions.

Head Polysemanticity. Some heads may serve multiple functions that our single-name system cannot fully capture.

Despite these limitations, we believe this taxonomy provides a valuable organizing framework for the field.

7 Conclusion

7.1 Summary of Contributions

This work introduces a unified naming framework for attention heads in modern transformer models. We provide a four-level depth model (Early/Middle/Late/Final), a stack-based functional taxonomy (14 stacks), canonical names for ∼80 attention head types, and a comprehensive cross-reference for historical terminology.

7.2 Adoption Guidelines

We recommend that researchers use canonical names in papers and documentation, include alternative names in parentheses when first mentioned, specify depth ranges when reporting head discoveries, and indicate primary stack membership for context. For example: "We identified an induction head (also called pattern head) at relative depth 0.35 in the Reasoning & Algorithmic stack."

7.3 Future Directions

This taxonomy opens several research directions:

Empirical Validation. Systematic studies validating head types across diverse models [9, 14].

Automated Detection. Tools for automatically identifying and classifying heads in new models [4].

Circuit Mapping. Using standardized names to build comprehensive circuit databases [13].

Architecture Design. Leveraging head taxonomy to design more interpretable models.

Safety Applications. Using head understanding to improve model alignment and safety [15, 2].

We hope this naming convention facilitates communication, enables replication, and provides structure to an expanding field.

A Alphabetical Cross-Reference Table

This table maps informal names found in the literature to our canonical naming convention. Format: `Literature name` → `(PREFIX) Canonical name`.

Literature Name	Canonical Name
algorithmic head	(M) Algorithmic continuation head
block head	(L) Block-structure head
boundary head	(E) Boundary head
carry head	(L) Carry head
char-level head	(E) Local pattern head
classification head	(E) Safety-classification head
code-block head	(L) Structural-block head
code-fence head	(L) Structural-block head
code-formatting head	(L) Structural-block head
code-structure head	(L) Block-structure head
coercion head	(L) Copy-suppression head
completion head	(F) Completion-stabilization head
consistency head	(F) Format-consistency head
continuation head	(M) Algorithmic continuation head
copy head	(L) Name-mover head / (M) Duplicate-token head
copy-suppression head	(L) Copy-suppression head
coreference head	(M) Coreference head
delimiter head	(E) Delimiter head
detection head	(E) Sensitive-content head
digit head	(M) Digit head
duplicate token head	(M) Duplicate-token head
entity head	(M) Entity head
explanation head	(M) Explanation head
fact head	(M) Fact head
fence head	(L) Structural-block head
final-layer head	(F) Format-consistency head
function head	(L) Formula-structure head
hallucination-suppression head	(L) S-inhibition head
hazard head	(E) Hazard-topic head
indentation head	(M) Indentation head
inhibition head	(L) S-inhibition head
instruction head	(E) Instruction head
intent head	(M) Task-mode head
JSON-format head	(L) Output-schema head
key-value head	(L) Key–value pairing head
list head	(L) List-structure head
local-pattern head	(E) Local pattern head
markdown head	(L) Structural-block head / (L) List-structure head
meta-cot head	(F) Meta-CoT head
mode head	(M) Mode-switch head

Continued on next page

Literature Name	Canonical Name
mover head	(L) Name-mover head
name mover head	(L) Name-mover head
narrative head	(M) Narrative style head
object head	(L) Key–value pairing head
operator head	(M) Operator head
output-format head	(L) Output-schema head
output-schema head	(L) Output-schema head
output-specification head	(F) Output-specification head
paren-matching head	(L) Paren-matching head
pattern head	(E) Local pattern head / (M) Induction head
persona head	(L) Persona head
polite head	(L) Politeness head
previous-token head	(E) Previous-token head
prompt head	(E) System-prompt head
python head	(L) Structural-block head
quoting head	(L) Structural-block head
rag-routing head	(F) Implicit-RAG routing head
reasoning head	(F) Reasoning-mode head
redirect head	(F) Redirect head
reference head	(E) Reference head
refusal head	(F) Refusal head
repetition head	(M) Duplicate-token head
retrieval head	(M) Schema retriever head
risk head	(E) Hazard-topic head
schema head	(M) Schema retriever head
scope head	(L) Scope head
sectioning head	(L) Sectioning head
self-description head	(L) Self-description head
semantic head	(L) Focus head
sensitive-content head	(E) Sensitive-content head
skip-trigram head	(M) Skip-trigram head
steering head	(L) Policy-enforcement head / (F) Reasoning-mode head
style head	(M) Tone head / (M) Narrative style head
suppression head	(L) Copy-suppression head
system head	(E) System-prompt head
task head	(M) Task-mode head
token-type head	(M) Token-type head
tone head	(M) Tone head
tone-softening head	(F) Tone-softening head
toxicity head	(E) Toxicity head
tracking head	(M) State-tracking head
translate head	(M) Entity head / (M) Fact head
whitespace head	(E) Whitespace-structure head
XML head	(L) Output-schema head
YAML head	(L) Output-schema head

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Andy Arditi, Oscar Obeso, Aaquib Kreutzer, Alex Rager, Eric Jenner, Esben Prakash, Nora Belrose, and Alex Turner. Refusal in llms is mediated by a single direction. *arXiv preprint*, 2024.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *OpenAI Blog*, 2023. URL https://openai.com/research/language-models-can-explain-neurons-in-language-models.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

[7] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[8] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[9] Daking Rai, Yilun Lee, Shi Feng Xuan, Leif Yao, Jeffrey Kwan, Eric Mitchell, and Chelsea Finn. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

[10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al.

Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

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

[14] Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Attention heads of large language models: A survey. *Patterns*, 6(2):101176, 2025. doi: 10.1016/j.patter.2025.101176.

[15] Andy Zhou et al. Refusal falls off a cliff: How safety alignment fails in reasoning? *OpenReview*, 2025. URL https://openreview.net.