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 [10], 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 [5, 6]. 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 [8, 13]. 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 [5, 12]. **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 [4, 9, 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 [12, 6]. 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* $\times$ *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 $\rightarrow$ Middle $\rightarrow$ Late $\rightarrow$ 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.

5.7 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.

5.8 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.

5.9 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.

5.10 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.

5.11 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.

5.12 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.

5.13 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.

5.14 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.

6 Discussion

6.1 Cross-Stack Patterns

Across architectures, consistent patterns emerge [8, 13]. 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 [11]. 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 [4], LLaMA models show strong instruction-following head patterns [9], and safety-tuned models have more pronounced safety stack heads [7, 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 [8, 13].

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

Circuit Mapping. Using standardized names to build comprehensive circuit databases [12].

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

Safety Applications. Using head understanding to improve model alignment and safety [14, 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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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