ChatGPT

Explainability in Large Language Models (LLMs): Layer Functions, Interpretability Techniques, and Tools

Introduction

Large language models (LLMs) have achieved remarkable capabilities, but their **internal workings remain largely opaque**, raising concerns about reliability and trust. As Zhao *et al.* (2023) note, "internal mechanisms are still unclear and this lack of transparency poses unwanted risks… understanding and explaining these models is crucial" [1]. Researchers are therefore actively studying *how* information is processed across transformer layers and developing interpretability methods to **probe** and **diagnose** LLM behavior. This report reviews recent (2024–2025) findings on *layer-specific functionality*, interpretability techniques/diagnostics, and modern tools for exploring LLM internals, highlighting progress and future challenges.

Layer-Specific Functionality in Transformers

Recent analyses reveal that **not all layers contribute equally** to an LLM's abilities. Zhang *et al.* (2024) use Shapley-value ablation to identify "cornerstone" layers whose removal causes performance to collapse. These tend to be **early layers**: ablating a single early transformer block often degrades accuracy to near random [2]. In contrast, several studies find that many later layers can be **pruned** with little effect. Gromov *et al.* (2024) show that deep layers in large LLMs (e.g. Llama-2-70B) are often *"unreasonably ineffective"*: the model tolerates removal of up to half its layers before performance drops substantially [3]. Similarly, He *et al.* (2024) find that *attention* sub-layers can be heavily pruned – dropping 50% of the attention layers in Llama-2-70B still yields near-original performance [4]. (Interestingly, in their pruning study, removing MLP sub-layers **hurts** performance, whereas attention layers are largely redundant.) These results suggest a pattern: **early/mid transformer layers carry much of the critical knowledge**, while many deeper layers may add only marginal or redundant transformations [2] [4].

How does the *type* of layer (attention vs. MLP) or its depth relate to its role? He *et al.* quantify layer importance by measuring the cosine similarity between a layer's input and output: high similarity indicates little change (redundancy) [5] [6]. They find many attention layers exhibit near-identity behavior and can be skipped [4]. In practice, this means **an LLM may store core knowledge in MLP layers or residual streams**, using attention primarily for propagation. Independent evidence comes from probing internal representations. Liu *et al.* (2024) probe Llama-2 with a lexical-semantic task: **lower layers encode lexical meaning**, whereas top layers (responsible for next-token prediction) carry weaker semantics [7]. This is opposite to BERT-like models, where semantics accumulate toward the top. Lei & Cooper (2025) further chart how *knowledge is organized geometrically across depth*: middle layers embed **continuous, overlapping attribute spaces** (e.g. chemical properties) that support indirect recall, while deeper layers "sharpen categorical distinctions" and inject linguistic context [8]. In sum, **mid-layers appear to house rich semantic manifolds**, whereas final layers refine the output distribution.

These findings are consistent with earlier NLP insights (e.g. Tenney *et al.* 2019 on BERT) that **lower layers tend to capture surface/syntactic information and word meanings, while higher layers focus on task-specific or contextual processing**. The emerging picture is that **each transformer block plays a distinct role**: early blocks build up basic lexical and syntactic features; middle blocks integrate and encode facts and concepts; later blocks adjust for prediction and context. However, much remains unknown about *specific* functions (e.g. which heads detect particular patterns or how MLP non-linearities specialize). Researchers continue to probe layer roles with targeted experiments and ablations.

Interpretability Techniques and Diagnostic Methods

Feature attribution and saliency methods. A common approach is to assign credit to inputs or internal units for an output. Techniques from vision (gradient saliency, integrated gradients, DeepLIFT) are adapted to LLMs. For example, **Integrated Gradients (IG)** and related path-attribution methods rank input tokens by influence on a prediction. Sanyal & Ren (2021) introduced a "discretized IG" for language models, and later work extends captum-based methods to sequence generation [9]. **LIME/SHAP** style methods also apply: SHAP (Shapley-based) and LIME (local surrogate) can explain word-level contributions to a given answer. These tools work *post hoc* on a trained LLM to highlight which words, neurons, or layers drive a decision. For example, the Inseq toolkit provides integrated gradients and other explainers for encoder-decoder and decoder-only models [9]. It can visualize token-level attribution for generation tasks (see Fig. 1 in Inseq docs). However, these methods can produce noisy or unstable explanations, especially with large vocabularies and context dependence. Recent work (Friedman *et al.*, 2023) cautions about **interpretability illusions**: simplified explanations (e.g. via PCA or reduced representations) may not faithfully reflect the full model's behavior, especially out-of-distribution [10].

Probing and probing classifiers. Diagnostic classifiers ("probes") are trained on layer activations to detect linguistic features (POS tags, syntax trees, semantics) [11]. By checking which layer's representations best support a given probe, one infers *what is encoded where*. For example, early transformer layers usually yield better probes for syntactic tasks, whereas deeper layers (or final representations) might excel at semantic or task-specific probes. Probing has revealed that **factual or lexical content tends to be linearly decodable from intermediate layers**, supporting the notion of distributed knowledge [8] [7]. However, probes can be overconfident: they may "read out" information not actually used by the model in generation. Thus, probing results must be interpreted cautiously (seeing what *could* be extracted, not necessarily what the model *uses*).

Attention and activation analysis. Attention weights are often visualized to identify which tokens attend to which. Early work (BERTology) showed some heads focus on syntax (e.g. attending to heads for noun phrases or coreference) [12]. While attention maps alone are not explanation-proof (they do not uniquely determine output), systematic analysis of attention patterns can highlight linguistic structure. Another tactic is to examine single neurons or directions in the activation space. Mechanistic interpretability frames neurons as **feature detectors**. For instance, one can search for a "French text" neuron that responds to French inputs [11]. Circuits combine heads and neurons: a famous example is the **induction head circuit** for copy/recurrence, discovered by Elhage *et al.* (2021) [13]. Such circuit discoveries involve identifying chains of attention-head activations that implement algorithmic functions. More generally, researchers use causal interventions (ablations, "path patching" of activations) to test hypotheses about circuits: e.g. setting one head's output to zero and observing the effect on final output. Causal scrubbing and counterfactual experiments help link internal components to behavior.

Evaluation and diagnostics. Several metrics assess interpretability quality. **Completeness/fidelity** measures whether the explanations capture the full model behavior (e.g. does a subset of features suffice to reproduce the output). **Faithfulness** checks if importance scores correlate with actual effect (by perturbing inputs or neurons). **Plausibility** evaluates how human-understandable the explanation is. These are active research topics (see surveys by Zhao *et al.* 2023 and others). To diagnose specific issues, specialized tests exist: e.g. probing for bias (Fairness metrics), robustness (adversarial tests), or consistency (AuditLLM checks for stable answers under paraphrased queries [14]). Overall, a combination of qualitative and quantitative diagnostics is used to build confidence in LLM behavior.

Tools for Exploring LLM Internals

Researchers have developed numerous open-source tools to inspect LLMs at various levels. Prominent examples include:

- **TransformerLens (formerly EasyTransformer)**: A Python library by Neel Nanda for *mechanistic interpretability*. TransformerLens lets you load many open models (GPT-2, Llama, etc.) and **hook into any hidden activation or weight matrix** during the forward pass [15]. Users can cache activations, modify them (e.g. ablate a head), and re-run the model to study causal effects. The design emphasizes rapid experimentation: as Nanda writes, it "exposes the internal activations" and enables short feedback loops for circuit analysis [15].

- **Ecco**: A toolkit by Jay Alammar that provides **interactive visualizations** in Jupyter notebooks. Ecco displays attention patterns, token embeddings, and prediction trajectories for GPT-2, BERT, T5, etc., helping users "explain, analyze, and visualize" model behavior [16]. It is user-friendly for educators and researchers to inspect what a language model "sees" when processing a sentence.

- **Inseq**: A PyTorch-based interpretability toolkit for sequence models. It implements **feature attribution** (integrated gradients, saliency, attention attribution, etc.) for encoder-decoder and decoder-only transformers [9]. Inseq lets users easily compute and visualize token-level attributions for generation tasks. It leverages Captum's methods (integrated gradients, DeepLIFT, etc.) and is designed to "democratize access" to these analyses [17] [9].

- **NeuronViewer (OpenAI)**: A web-based interactive tool for exploring neuron activations and explanations in GPT-2/3 models. It allows filtering neurons by activation patterns, labeling high-activating examples, and testing interventions on a neuron.

- **TransformerViz**: A browser-based app to visualize attention and token flows through a transformer's layers, making it easier to track how information propagates.

- **InterpretML**: While not specific to LLMs, Microsoft's InterpretML provides a unified framework of explainability methods (both "glassbox" models like Generalized Additive Models and "blackbox" tools like LIME and SHAP) [18]. Practitioners can use it to explain any model, including text classifiers, by attributing feature importance.

- **Other frameworks**: Tools like **Captum** (PyTorch's interpret library), **LIME/SHAP** packages, and **AllenNLP Interpret** offer built-in explanation methods (saliency maps, counterfactuals) for neural

models. Platforms like the Learning Interpretability Tool (PAIR/Google) and **Phoenix** or **Floom** provide end-to-end LLM evaluation, including interpretability dashboards. Rapidly emerging are cloud services (Langchain's **LangSmith**, Coralogix, Datadog) that add visualization layers for LLM deployments. Many of these tools incorporate techniques such as attention visualization, token attribution, or residual-stream inspection to shed light on outputs.

Each tool has tradeoffs (ease-of-use vs flexibility, small model vs large model support). Together, they offer a growing ecosystem for LLM interpretability, enabling both research experiments (e.g. finding circuits) and applied debugging (e.g. checking bias or performance bottlenecks).

Key Progress, Limitations, and Future Directions

Progress: In 2024–2025, interpretability research has advanced our understanding of LLM internals significantly. Researchers have identified "cornerstone" components, quantified layer redundancies, and begun mapping semantic content to specific layers [2] [8]. New methodologies (Shapley analyses, logit/tuned lenses, causal scrubbing) have been introduced to audit model structure. The tool landscape has matured: TransformerLens and similar libraries make mechanistic experiments much easier, and attribution toolkits like Inseq enable widespread application of gradient-based explanations. Survey papers and benchmarks are organizing the field, and open-source model analyses (e.g. "circuits" discovered in GPT-2/3) are illuminating emergent patterns. There is also progress on standards and metrics for interpretability (completeness, fidelity measures) and on transparency for LLM safety (bias audits, output consistency tests).

Limitations: Despite these gains, serious challenges remain. *Scalability and trust:* Most mechanistic insights come from small or medium models (GPT-2, smaller LlaMA); it is unclear if circuits or feature representations found there directly scale to multi-billion-parameter models. *Partial explanations:* Attribution methods (like saliency) often conflict and may be misleading. As Friedman *et al.* warn, simplified explanations might be faithful in-distribution but fail out-of-distribution, giving a false sense of understanding [10]. *Polysemanticity:* Many neurons appear to respond to multiple unrelated features, making it hard to assign a single "role" to an activation. *Evaluation gaps:* There is no consensus on how to rigorously evaluate an explanation, so claims of interpretability can be subjective. *Tool maturity:* Many analysis tools are research prototypes (Jupyter-based or offline). Integration into production or real-time debugging is still developing.

Future Directions: The field is moving toward more **holistic and automated interpretability**. One direction is discovering **higher-level concepts** or "naturally emergent" features in LLMs via unsupervised methods, rather than manually probing one feature at a time. Another is scaling mechanistic analysis through partial automation (e.g. "Circuit Book" catalogs, automated search for circuits). Researchers are also exploring **hybrid models**: designing model architectures with interpretability in mind, or embedding explainers into the training process (self-explaining models). Evaluation of explanations will become more rigorous, possibly borrowing techniques from causal inference. On the tool side, we expect richer interactive platforms (e.g. combining Vector-DB style search with activation analysis) and integration of explanation pipelines into development frameworks. Finally, as LLMs expand to multimodal and continual learning settings, new interpretability challenges will arise (e.g. how visual and textual representations interplay).

In summary, **significant strides** have been made in peering inside LLMs, revealing intriguing layer-wise behaviors and offering ever-more-powerful analysis tools. Nevertheless, interpretability is far from "solved."

Ongoing research must address the **reliability** of explanations, scale them to the largest models, and translate findings into safer, more controllable AI systems. The coming years will likely see deeper collaboration between theory (mechanistic circuits, information geometry) and practice (observability platforms, regulatory requirements) to realize the promise of truly transparent language AI.

Sources: Key findings are drawn from recent literature on transformer interpretability [2] [4] [8] [7] and from documentation of open-source tools [15] [17] [9] [18]. Additional context and survey perspectives come from large-scale studies [1] [10].

---

[1] [2309.01029] Explainability for Large Language Models: A Survey
https://arxiv.org/abs/2309.01029

[2] [12] Investigating Layer Importance in Large Language Models
https://arxiv.org/html/2409.14381v1

[3] The Unreasonable Ineffectiveness of the Deeper Layers
https://arxiv.org/html/2403.17887v1

[4] [5] [6] What Matters in Transformers? Not All Attention is Needed
https://arxiv.org/html/2406.15786v4

[7] aclanthology.org
https://aclanthology.org/2024.findings-acl.866.pdf

[8] Layerwise Recall and the Geometry of Interwoven Knowledge in LLMs
https://arxiv.org/html/2502.10871v2

[9] [17] GitHub - inseq-team/inseq: Interpretability for sequence generation models
https://github.com/inseq-team/inseq

[10] [2312.03656] Interpretability Illusions in the Generalization of Simplified Models
https://ar5iv.labs.arxiv.org/html/2312.03656

[11] [13] A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models
https://arxiv.org/html/2407.02646v1

[14] AuditLLM: A Tool for Auditing Large Language Models Using Multiprobe Approach
https://arxiv.org/html/2402.09334v2

[15] TransformerLens Documentation
https://transformerlensorg.github.io/TransformerLens/

[16] GitHub - jalammar/ecco: Explain, analyze, and visualize NLP language models. Ecco creates interactive visualizations directly in Jupyter notebooks explaining the behavior of Transformer-based language models (like GPT2, BERT, RoBERTA, T5, and T0).
https://github.com/jalammar/ecco

[18] InterpretML: Another Way to Explain Your Model | by Noga Gershon Barak | TDS Archive | Medium
https://medium.com/data-science/interpretml-another-way-to-explain-your-model-b7faf0a384f8