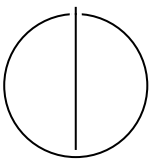
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Knowledge Boundaries of LLMs in Specialized Domains

Ren Jeik Ong



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

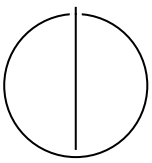
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Knowledge Boundaries of LLMs in Specialized Domains

Erforschung der Wissensgrenzen von LLMs in spezialisierten Domänen

Author:	Ren Jeik Ong
Examiner:	Prof. Dr. Georg Groh
Supervisor:	Tobias Eder
Submission Date:	07.11.2025



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 07.11.2025 Ren Jeik Ong

Acknowledgments

I would like to thank Tobias who has been supporting me and giving me directions in completing the most enjoyable topic. Without him, I would not have explored this interesting topic and deep knowledge in this area. I'm also grateful to have Prof. Georg and members of Chair of Social Computing for constant support and trust. On top of that, I would like to say thank you to Edoardo Mosca for giving me some advice in his field of knowledge and expertise.

Last but not least, I would like to thank my close friends and family members for always giving me moral and mental support.

Abstract

In the context of Large Language Model (LLM)s, the concept of a knowledge boundary refers to the dividing line between regions of known and unknown knowledge. This boundary is not limited to a binary threshold of correct versus incorrect output but encompasses the extent to which the model analyzes input and produces responses that are truthful and faithful. Despite the progress of state-of-the-art (SOTA) LLMs, some of which surpass human performance in domain-specific tasks, the assurance of consistent truthfulness and faithfulness remains unresolved.

This thesis introduces **layerwise entropy**, a metric designed to capture the internal representations of LLMs and evaluate the degree to which each layer is effectively utilized during inference. The method integrates entropy measurements across hidden states, from raw contextual embeddings through intermediate layers, enabling the detection of instability within the model.

By decoding raw score logits and applying inference across multiple architectures, including *Transformer*, *Mamba*, and *Linear Input-Varying (LIV)-Synthesis of Tailored Architectures (STAR)* models, this research investigates the entropy distribution across layers. The analysis highlights differences between regions of known and unknown knowledge and explores how such differences may signal hallucinations. Ultimately, layerwise entropy is proposed as a step toward mitigating hallucinations by deepening our understanding of the internal dynamics of LLMs.

Contents

1. Introduction

LLMs have become a central paradigm in Natural Language Processing (NLP), demonstrating impressive capabilities in translation, summarization, question answering, and long-form text generation. Their effectiveness stems from large-scale training data and sophisticated neural architectures. Yet, as their performance continues to expand, questions remain about the scope and limits of their knowledge. In particular, the concept of a *knowledge boundary* is essential to understanding how and why LLMs produce accurate outputs in some cases but unreliable or fabricated ones in others.

Knowledge boundary of an LLM can be described as the dividing line between regions of known and unknown knowledge (Cheng, T. Sun, X. Liu, et al. 2024). Unlike traditional systems with explicitly defined knowledge bases, LLMs rely on high-dimensional representations learned from vast corpora. This makes their boundaries less transparent. More importantly, these boundaries are not strictly binary. Instead, they reflect a spectrum of uncertainty, in which a model may generate text that is coherent and convincing but lacks factual accuracy (Ji, Lee, Frieske, et al. 2023). This phenomenon, often termed *hallucination*, highlights the limitations of relying solely on surface-level fluency as a measure of reliability.

Although SOTA models achieve or even surpass human performance in some specialized domains, consistent truthfulness and faithfulness remain unresolved challenges. Nonetheless, current evaluation practices, such as benchmark accuracy or human preference ratings, cannot fully capture the internal processes that govern reliability. Therefore, there is a need for probing the hidden states of LLMs, offering insights into how knowledge is processed and represented during inference.

This thesis proposes *layerwise entropy* as such a method. Layerwise entropy measures uncertainty across the hidden states of an LLM, from raw contextual embeddings through intermediate layers. By analyzing entropy at these stages, it becomes possible to assess whether each layer is effectively utilized and to detect potential instability within the model. Layerwise entropy provides an internal, structural perspective. This distinction matters because hallucinations may arise not only in final outputs but also from instability within specific layers.

To demonstrate its applicability, layerwise entropy is evaluated across multiple architectures, including Transformer (Vaswani, Shazeer, Parmar, et al. 2023), Mamba (Gu and Dao 2024), and LIV-STAR (Thomas, Parnichkun, Amini, et al. 2024) models. By

comparing entropy distributions in regions of known and unknown knowledge, the study examines whether patterns of internal uncertainty align with hallucination tendencies. This analysis contributes both a methodological tool for probing LLMs and a practical step toward reducing hallucinations.

Ultimately, the thesis incorporates layerwise entropy as a means to better understand knowledge boundaries and model reliability. As LLMs are increasingly deployed in sensitive domains such as law, medicine, and education, addressing hallucination is not merely a technical challenge but also a practical necessity. By offering a framework for exploring internal uncertainty, this work seeks to bridge the gap between observable outputs and the deeper dynamics that shape knowledge expression in LLMs.

The source code for layerwise entropy is available as open source on GitHub.[1]

[1] `https://github.com/ongxx107/layerwise-entropy`

2. Background

2.1. Large Language Models

LLMs represent a major advancement in NLP. Their ability to capture linguistic patterns and generate coherent text stems from large-scale training on diverse corpora combined with sophisticated neural architectures. The most widely adopted framework is the Transformer, which leverages self-attention mechanisms to efficiently process long sequences and learn contextual dependencies (Vaswani, Shazeer, Parmar, et al. 2023).

While Transformers dominate current applications, alternative architectures have also been proposed. Recent developments such as Mamba (Gu and Dao 2024) and Mamba-hybrid (Glorioso, Anthony, Tokpanov, et al. 2024) models integrate sequence modeling innovations with recurrent and state-space mechanisms (Gu, Goel, and Ré 2022). These architectures aim to address some of the scalability and efficiency challenges of Transformers while preserving high levels of expressivity.

2.1.1. Transformer Architecture

Transformer is built on self-attention mechanisms, positional encodings, and feed-forward networks. Self-attention allows each token to attend to every other token in a sequence, enabling the model to capture both local and global dependencies. Multi-head attention further enriches this process by learning diverse contextual relationships in parallel. Residual connections and layer normalization contribute to stable training, while stacked layers yield increasingly abstract representations.

2.1.2. Mamba and Mamba-Hybrid Architectures

Mamba is a sequence model architecture based on selective state-space mechanisms designed to balance efficiency with expressive power. Unlike Transformers, which rely heavily on quadratic attention mechanisms, Mamba models operate with linear-time complexity, making them more efficient for long sequences. The Mamba-hybrid architecture combines State Space Model (SSM) modeling with Transformer-like layers,

leveraging the strengths of both approaches. These models provide a useful contrast for evaluating entropy across diverse architectures.

2.1.3. LIV and STAR: A Unified Design-Search Framework

LIV systems cast sequence layers as linear operators whose coefficients are modulated by input-derived features, decoupling *featurization* from operator *structure* (Thomas, Parnichkun, Amini, et al. 2024). Under this view, attention, linear-attention, gated convolutions, and state-space or recurrent layers appear as specific structural choices within the same design space. Building on this, STAR framework composes backbones from LIV building blocks via a hierarchical search over featurizers, operator structures, and inter-layer composition rules.

By exploring sharing strategies and non-sequential connections, STAR discovers potential models that match or surpass strong Transformer and hybrid baselines at comparable scale while optimizing quality-efficiency trade-offs.

2.2. Knowledge Boundaries in LLMs

Despite their success, LLMs remain limited by the scope of their learned knowledge. A *knowledge boundary* describes the point at which the model transitions from generating factually grounded responses to producing uncertain or fabricated information (Cheng, T. Sun, X. Liu, et al. 2024). Unlike a strict binary threshold, this boundary reflects a continuum in which responses may vary from accurate to partially correct or entirely fabricated (Huang, Yu, Ma, et al. 2025; Ji, Lee, Frieske, et al. 2023).

Understanding knowledge boundaries is critical for assessing model reliability. When boundaries are not well-defined, LLMs risk generating hallucinations. Outputs that are coherent and fluent yet lack factual basis. This phenomenon undermines trust in LLMs, especially in high-stakes applications such as law, medicine, and education.

2.2.1. Truthfulness, Faithfulness, and Hallucination

Two important dimensions in evaluating knowledge boundaries are *truthfulness* and *faithfulness*. Lin et al. define truthfulness as the factual accuracy of a model's outputs relative to external reality (S. Lin, Hilton, and Evans 2022), while Ji et al. define faithfulness as the consistency of outputs with respect to the model's internal reasoning or input data. Failures along these dimensions often manifest as hallucinations, where the model generates plausible-sounding but incorrect content.

Hallucination arises partly from the probabilistic nature of language generation. Because models are trained to predict the next token based on likelihood rather than

truth, they may produce semantically coherent but factually invalid sequences. This motivates the need for tools that probe internal states rather than relying exclusively on surface-level correctness.

2.3. Uncertainty Estimation in LLMs

Uncertainty estimation is central to understanding the reliability of LLM outputs. Since these models generate responses probabilistically, evaluating confidence and distinguishing between well-grounded and speculative outputs is a critical challenge. Conventional evaluation metrics, such as accuracy or BLEU scores, measure performance but fail to capture deeper model uncertainty (Mathur, Baldwin, and Cohn 2020). As a result, recent work quantifies uncertainty either *on the vocabulary simplex* via the next-token distribution induced by model logits (output-based/predictive), such as semantic entropy (Kuhn, Gal, and Farquhar 2023), or *within the model's internal activations* by measuring dispersion directly in hidden states (representation-based), such as Singular Value Decomposition (SVD) score, which uses eigenvalue analysis of the covariance matrix (Sriramanan, Bharti, Sadasivan, et al. 2024).

2.3.1. Semantic Entropy

Semantic entropy is an output-based method for quantifying uncertainty. Instead of focusing solely on raw token probabilities, semantic entropy evaluates the variability across multiple generations produced by the same prompt. The core idea is that if repeated generations vary significantly in meaning, the model is uncertain about the response.

Formally, semantic entropy measures the distribution of meanings across sampled text outputs rather than the surface-level token distributions. This allows it to capture uncertainty in a way that aligns more closely with user expectations, since many forms of hallucination emerge when models generate multiple semantically divergent yet plausible responses. However, semantic entropy requires repeated sampling, which increases computational cost and does not provide insights into the structural origins of uncertainty within the model.

2.3.2. Layerwise Entropy

Inspired by (S. Chen, M. Xiong, J. Liu, et al. 2024) and the entropy-decoding implementation in (OptimLLM 2024), layerwise (predictive) entropy is not an output-based metric, but rather an (internal) output-space metric. It is defined on the probability simplex over the vocabulary: at each layer, the hidden state is projected through the

language head to obtain a next-token distribution and its Shannon entropy is computed. In contrast, representation-based approach derives from dispersion directly in hidden-state space and quantifies internal variability rather than how predictive a layer is about the next token.

The primary advantage of layerwise entropy is its ability to reveal whether each layer is effectively utilized during inference. High entropy may indicate instability or underutilization, signaling potential points where hallucinations could arise. Unlike semantic entropy, layerwise entropy does not require repeated sampling of outputs, making it more efficient and more directly tied to the internal structure of the model.

2.3.3. Other Approaches to Uncertainty

In addition to semantic and layerwise entropy, other methods have been proposed to capture uncertainty in LLMs:

- **Calibration methods** attempt to align model confidence with empirical accuracy, often through temperature scaling or post-hoc adjustments (Guo, Pleiss, Y. Sun, and Weinberger 2017). These improve probability estimates but do not explain internal representations.
- **Bayesian and ensemble methods** explore uncertainty by aggregating predictions from multiple models or parameter samples (Lakshminarayanan, Pritzel, and Blundell 2017). While theoretically grounded, these approaches are computationally expensive and challenging to scale to modern LLMs.

Each of these methods contributes to understanding model uncertainty but comes with trade-offs in efficiency, interpretability, and scalability. Within this landscape, layerwise entropy offers a structural, efficient, and interpretable approach to probing uncertainty inside LLMs, making it particularly suitable for studying knowledge boundaries.

2.4. Entropy and Information-Theoretic Measures

Entropy, first introduced by Shannon in the context of information theory (Shannon 1948), provides a fundamental measure of uncertainty in probability distributions. Entropy quantifies the average amount of information required to describe a random variable, offering a mathematical framework for reasoning about uncertainty and predictability. This concept has been widely adopted in machine learning and NLP, where language can be modeled as a probabilistic distribution over sequences of tokens.

2.4.1. Entropy in Early NLP Applications

Before the advent of deep learning, entropy played a significant role in classical NLP tasks. In n-gram language models, entropy is used to evaluate the predictability of words given their preceding context (S. F. Chen and Goodman 1999). Lower entropy indicated more confident predictions, while higher entropy signaled uncertainty about upcoming tokens. Similarly, entropy-based measures are applied to word sense disambiguation, machine translation, and part-of-speech tagging, where they provided insights into model confidence and data sparsity.

2.4.2. Entropy in Neural Language Models

With the transition to neural architectures, entropy retained its relevance as a diagnostic tool. Token-level entropy, derived from the probability distribution over the next-token prediction, became a common measure of model uncertainty (Malinin and Gales 2021).

More recently, semantic entropy has been introduced to move beyond surface probabilities by evaluating diversity across multiple generated outputs. This approach aligns entropy more closely with human perceptions of uncertainty, as it reflects semantic variability rather than token overlap. Yet, semantic entropy requires repeated sampling, limiting efficiency and interpretability.

2.4.3. Toward Layerwise Entropy

Building on this trajectory, layerwise entropy extends the application of entropy into the internal structure of LLMs. Instead of focusing solely on outputs, it quantifies uncertainty in hidden states across layers. This shift reflects a broader movement in NLP toward probing internal representations for interpretability and reliability.

By connecting information-theoretic principles with modern deep architectures, layerwise entropy provides a structural view of uncertainty, offering both diagnostic insights and practical pathways for reducing hallucinations. Readers interested in the formal derivations of entropy and variational entropy are referred to Section 4.5.

2.5. Evaluation of LLM Reliability

Evaluating the reliability of LLMs is an ongoing challenge in NLP. Traditional metrics, such as accuracy, precision, recall, and F1-score, are effective for classification tasks but insufficient for open-ended text generation, where outputs may be factually incorrect yet linguistically fluent. As LLMs are increasingly applied to knowledge-intensive

domains, the ability to measure not only performance but also reliability has become essential.

Benchmark datasets have played a central role in model evaluation. Datasets such as General Language Understanding Evaluation (GLUE) (Wang, Singh, Michael, et al. 2019), SuperGLUE (Wang, Pruksachatkun, Nangia, et al. 2020), and Massive Multitask Language Understanding (MMLU) (Hendrycks, Burns, Basart, et al. 2021) test reasoning, factual recall, and generalization across diverse domains. While these benchmarks provide useful comparisons between models, they primarily measure correctness relative to static datasets and fail to capture nuanced issues such as hallucination or calibration of uncertainty. Similarly, automated text comparison metrics, including Bilingual Evaluation Understudy (BLEU) (Papineni, Roukos, Ward, and Zhu 2002), Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (C.-Y. Lin 2004), and Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie 2005), are widely used for translation and summarization tasks but fall short when assessing truthfulness in free-form generation, since semantically incorrect outputs may still achieve high lexical overlap with reference texts.

Human evaluation remains the most reliable method for assessing faithfulness and truthfulness in open-ended tasks. Human judges can assess whether responses are factually correct, logically consistent, and contextually appropriate. However, human evaluation is costly, subjective, and difficult to scale across large datasets. Furthermore, annotator disagreement highlights the complexity of defining reliability in tasks where multiple answers may appear plausible.

Recent advances attempt to address these shortcomings by developing specialized benchmarks and metrics. Datasets such as TruthfulQA (S. Lin, Hilton, and Evans 2022) explicitly test a model's tendency to produce truthful rather than misleading answers, while other works propose fact-checking pipelines that compare model outputs against structured knowledge bases (Mountantonakis and Tzitzikas 2023). Another line of work explores calibration, which aligns a model's predicted confidence with its empirical accuracy, as demonstrated by Guo et al. Yet, despite these developments, evaluation methods remain limited in their ability to explain why models succeed or fail. This gap motivates methods that probe internal model representations, complementing surface-level evaluation with deeper insights into the sources of unreliability.

2.6. Interpretability and Internal Representations

The interpretability of LLMs is critical for understanding how they process and represent knowledge. Neural networks operate through distributed representations in high-dimensional vector spaces, making their internal decision-making opaque. As

models grow in scale and complexity, interpretability research seeks to uncover the mechanisms by which information is encoded, transformed, and expressed.

Hidden states, which capture intermediate representations across model layers, provide an entry point for interpretability. Probing studies have demonstrated that specific layers encode syntactic structures, semantic roles, or factual associations, suggesting that knowledge in LLMs is distributed across depth (Hewitt and Manning 2019; Tenney, Das, and Pavlick 2019).

Attention mechanisms have also been studied as a lens for interpretability. Visualization of attention weights reveals how tokens attend to each other, often uncovering linguistic or semantic relationships (Clark, Khandelwal, Levy, and Manning 2019). However, attention alone does not fully explain model reasoning, as high attention weights do not necessarily imply causal influence on the final prediction (Jain and Wallace 2019). Consequently, researchers increasingly turn to hidden-state analysis and representation probing to gain a more faithful understanding of internal processes.

Interpretability methods extend beyond visualization. Linear probing, clustering, and causal interventions attempt to identify what types of information are represented at different stages of computation. More recently, techniques such as representation similarity analysis have been used to compare model embeddings with human-annotated linguistic structures (Abnar, Beinborn, Choenni, and Zuidema 2019). These methods demonstrate that internal states hold meaningful information, but they also highlight instability: representations may vary across runs, prompts, or training settings.

The study of interpretability is closely tied to reliability. If hallucinations and boundary failures are reflected in unstable or underutilized representations, then probing internal states can offer predictive signals of when a model is likely to fail. This perspective motivates the development of entropy-based approaches such as layerwise entropy, which quantify uncertainty directly within hidden states. By connecting interpretability with uncertainty estimation, such approaches provide a structural path toward reducing hallucination and improving trust in LLMs.

2.7. Hallucination in LLMs

One of the most pressing challenges in deploying LLMs is their tendency to generate hallucinations. A hallucination occurs when a model produces fluent, coherent, and seemingly confident text that is factually incorrect, irrelevant, or unsupported by available data. This phenomenon arises because LLMs optimize for likelihood of text continuation rather than factual grounding. As a result, models may prioritize fluency over truthfulness, leading to outputs that are persuasive but unreliable.

2.7.1. Taxonomy of Hallucinations

As noted in the papers by Ji et al. and Huang et al., hallucinations can be classified into two broad categories:

- **Intrinsic hallucination** is generated output that contradicts or misrepresents the source input. For example, in summarization, attributing a statement to the source text that is not present.
- **Extrinsic hallucination** is generated output that is unsupported or unverifiable against the provided input or context. It often arises in Question Answering (QA) or dialogue when the model supplies details that the input does not justify, regardless of whether those details are true in the real world.

This taxonomy highlights the multiple ways hallucinations manifest across tasks, complicating evaluation and mitigation efforts.

2.7.2. Causes of Hallucination

In Ji et al.'s paper, several factors contribute to hallucinations in LLMs:

- **Training objectives:** LLMs are trained to predict the next token, not to verify factual correctness. This encourages fluent but potentially misleading generations.
- **Data limitations:** Training corpora contain noise, biases, and factual inaccuracies that may be reproduced or amplified by the model.
- **Generalization beyond training:** When queried about information outside its training distribution, a model may extrapolate incorrectly, producing outputs that are syntactically valid but factually false.
- **Decoding strategies:** Techniques such as beam search or nucleus sampling can exacerbate hallucinations by favoring tokens that maximize fluency without regard for factuality.

2.7.3. Hallucinations in High-Stakes Domains

The risks associated with hallucinations are magnified when LLMs are applied in sensitive areas such as law, medicine, and education. In legal contexts, fabricated case citations or statutes can mislead practitioners. In medicine, hallucinated treatment recommendations pose direct risks to patient safety. Even in educational settings, hallucinated facts may undermine learning and propagate misinformation. These risks make hallucination not merely a technical challenge but also an ethical and societal concern.

3. Overview

LLMs have rapidly become central to knowledge-intensive applications, yet their reliability in specialized domains remains uneven. As outlined in Chapter 2, LLMs can produce fluent but factually incorrect content when prompted beyond the scope of their learned knowledge. This work studies these *knowledge boundaries*, the regions where models transition from grounded to speculative or fabricated responses (Cheng, T. Sun, X. Liu, et al. 2024), with the aim of making such boundaries explicit, measurable, and actionable in downstream systems.

Despite growing interest in uncertainty estimation and interpretability, current practice is fragmented. Output-level metrics flag uncertainty after generation but provide limited insight into *where* inside the model uncertainty emerges (Kuhn, Gal, and Farquhar 2023). Conversely, representation-based interpretability reveals internal structure but is seldom tied to operational signals for boundary detection (Sriramanan, Bharti, Sadasivan, et al. 2024). In high-stakes settings such as medicine, law, and education, this gap leads to unreliable behavior: boundaries remain implicit, hallucinations go undetected, and systems lack principled mechanisms to defer, retrieve, or verify.

Thesis Contribution. This thesis develops a framework for identifying, measuring, and localizing knowledge boundaries of LLMs in specialized domains. Specifically, it (i) introduces *layerwise entropy*, which projects hidden states through the language head to form per-layer next-token distributions and computes their Shannon entropy, and (ii) relates this signal to truthfulness and faithfulness criteria, enabling principled deferral and mitigation strategies.

Scope and Assumptions. This study focuses on inference-time analysis of pretrained and fine-tuned LLMs. RLHF procedure design and large-scale deployment optimizations are beyond scope. Layerwise entropy is computed from single-pass hidden states without requiring repeated sampling.

Chapter Roadmap. Chapter 4 presents fine-tuning methodology, paraphrasing procedure, evaluation framework, Attention-Mamba-STAR architectures, and layerwise entropy probes. Chapter 5 describes implementation details, including data processing pipelines, model training setup, generation protocol, and supporting software

infrastructure. Chapter 6 reports experimental results, combining quantitative analyses with illustrative case studies. Chapter 7 discusses limitations and implications, and Chapter 8 concludes by summarizing the key findings and outlining future directions for integrating internal signals into the development of more trustworthy LLM systems.

4. Methods

4.1. Fine-Tuning and Parameter-Efficient Approaches

4.1.1. Overview

To adapt the base LLM to the specialized domain, Parameter-Efficient Fine-Tuning (PEFT) techniques are employed. Full model fine-tuning, while effective, is computationally intensive due to the vast number of parameters in contemporary LLMs. Instead, Low-Rank Adaptation (LoRA) and PEFT framework are adopted, both of which allow domain-specific adaptation with substantially reduced training overhead.

LoRA introduces low-rank decomposition into weight updates, restricting the number of trainable parameters while preserving representational capacity (Hu, Shen, Wallis, et al. 2021). Given a weight matrix $W \in \mathbb{R}^{d\times k}$, two low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are introduced such that the adapted weight becomes

$$W' = W + BA, \qquad r \ll \min(d, k).$$

The PEFT framework generalizes this principle by supporting multiple adapter-based strategies, enabling targeted fine-tuning of task-relevant modules. Together, these methods enable efficient adaptation while mitigating resource constraints.

Adapter Targeting and Capacity Pathing. Adapter placement is treated as a means to probe where domain knowledge is encoded. Targeting attention projections (e.g., W_Q, W_K, W_V) emphasizes interaction structure, whereas targeting Multi-Layer Perceptron (MLP) projections emphasizes feature transformation. Varying the LoRA rank r defines a controlled *capacity path*; monotone trends and diminishing returns along this path are interpreted as evidence about the degrees of freedom required for domain transfer.

Stability, Initialization, and Merge Invariance. To preserve the base function at the onset of training, low-rank updates are initialized near zero and optionally scaled by α/r. Because LoRA updates are additive, adapters can be merged into base weights at evaluation time without changing the function, separating representational changes from interface effects.

Compositionality and Regularization. For potential subdomains, the methodology accommodates parallel or serial adapter composition to examine interference vs. modularity. Conceptually, sparsity or orthogonality preferences on (A, B) constrain updates to low-dimensional, interpretable subspaces, reinforcing the structural prior.

4.1.2. Conceptual Rationale

We view LoRA/PEFT as imposing a structural prior that domain transfer can be realized through a low-dimensional perturbation of pretrained features. Varying the rank r traces a controllable capacity path, enabling attribution of performance changes to additional degrees of freedom rather than wholesale re-optimization. Because the perturbation BA is low-rank, these methods act as capacity control: they favor small, structured deviations that preserve the general competencies acquired during pretraining while specializing to domain-specific patterns.

Targeting particular submodules (e.g., attention projections vs. MLP projections) implicitly tests where domain knowledge is represented in interaction patterns or feature transformations, and thus helps localize the model's knowledge boundary. We further note an identifiability caveat: equivalent functions can arise from different parameterizations, so interpretations are constrained to comparisons under parameter-, Floating-Point Operation (FLOP)-, or training-budget-matched conditions.

4.2. Paraphrasing Procedure

4.2.1. Overview

Paraphrasing is employed to assess model robustness to linguistic variation while preserving semantic intent. Domain-specific questions are first compiled as a baseline input set. These are then paraphrased to introduce lexical, syntactic, and structural diversity without altering underlying meaning.

4.2.2. Pipeline

The procedure followed three stages:

1. **Generation:** Candidate paraphrases are generated using a transformer-based model under semantic similarity constraints.

2. **Filtering:** Low-quality outputs are eliminated based on grammaticality, fluency, and deviation from original meaning.

3. **Validation:** A final verification step ensured that retained paraphrases preserved domain intent while introducing genuine linguistic variation.

Transformation Taxonomy and Difficulty Control. Paraphrases instantiate a class T of meaning-preserving transformations, including lexical (synonymy, hypernymy), syntactic (voice alternation, clause reordering), discourse-level (co-reference restructurings), and formatting changes. To examine robustness gradients, paraphrases are stratified by semantic similarity bands, allowing analysis as similarity decreases while meaning remains constant.

Quality Controls and Artifact Mitigation. Proxies for faithfulness semantically and syntactically, together with diversity measures (e.g., edit distances or type-token ratios), are used to reduce redundancy while preserving label invariance. Paraphrases that introduce unintended cues (e.g., tense shifts that alter factual scope) are excluded. Borderline cases are subject to minimal adjudication that focuses strictly on meaning preservation.

4.3. Evaluation on Known and Unknown Questions

4.3.1. Overview

Evaluation is conducted on two complementary datasets. The *known set* consisted of the original test questions together with paraphrased versions of training-set questions, thereby assessing robustness to surface-level variation within the same semantic domain. The *unknown set* comprised novel queries specifically designed to lie outside the training data distribution, providing a measure of generalization to unfamiliar inputs. This division enables a systematic comparison of performance across both conditions. Evaluation metrics include macro-F1, accuracy, and uncertainty summaries, offering a comprehensive view of adaptation effectiveness.

4.3.2. Pipeline

The evaluation proceeds in two stages that target complementary goals. First, we report headline performance on the original test set only, using macro-F1 and accuracy. Constraining these metrics to the unmodified questions provides a consistent basis for comparing the fine-tuned model with the base model, which isolates the effect of fine-tuning without confounds from paraphrases or distribution shifts.

Second, we analyze model uncertainty across data regimes using layerwise entropy. For each intermediate layer, we compute predictive entropy on the known question set

and on the unknown question set. We then compare these uncertainty profiles between the fine-tuned model and the base model to examine whether fine-tuning reduces uncertainty on in-distribution inputs and how each model calibrates uncertainty on out-of-distribution (OOD) inputs. Together, the performance metrics on the original test set and the layerwise entropy summaries across known and unknown questions offer a focused view of effectiveness and generalization.

4.4. Attention, Mamba, and STAR Architectures

4.4.1. Overview

Two main architectural paradigms are examined: transformer-based models employing self-attention (Vaswani, Shazeer, Parmar, et al. 2023) and SSMs based on the Mamba framework (Gu and Dao 2024). In addition, we consider STAR design framework, which together provide a unified lens and automated methodology for composing efficient sequence-model backbones (Thomas, Parnichkun, Amini, et al. 2024).

4.4.2. Self-Attention Formulation

Self-attention computes pairwise token interactions to produce context-sensitive representations:

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

where Q, K, V denote the query, key, and value matrices, and d_k is the scaling factor.

4.4.3. Mamba (State-Space) Formulation

From SSMs (S4) to selective SSMs (S6). Let $x(t) \in \mathbb{R}$ denote a one–dimensional input signal and $y(t) \in \mathbb{R}$ the output, mediated by a latent state $h(t) \in \mathbb{R}^N$. A continuous SSM is

$$\begin{aligned} \dot{h}(t) &= A\,h(t) + B\,x(t), \\ y(t) &= C\,h(t), \end{aligned}$$

with learnable (A, B, C) and a per–channel step size Δ. Discretizing with a Zero-Order Hold (ZOH) maps $(\Delta, A, B) \mapsto (\bar{A}, \bar{B})$ via

$$\bar{A} = \exp(\Delta A), \qquad \bar{B} \;=\; (\Delta A)^{-1}\big(\exp(\Delta A) - I\big)\,\Delta B,$$

yielding the time–invariant recurrence for sequence data $x_{1:L}$:

$$h_t = \bar{A}\, h_{t-1} + \bar{B}\, x_t, \qquad y_t \;=\; C\, h_t.$$

Equivalently, the input–output map can be viewed as a global convolution $y = x * \bar{K}$ with kernel terms $\bar{K} = (C\bar{B},\, C\bar{A}\bar{B},\, C\bar{A}^2\bar{B}, \ldots)$, which enables parallel training via Fast Fourier Transform (FFT) while keeping recurrent inference linear in sequence length.

Mamba's selection mechanism (S6). A core limitation of Linear Time–Invariant (LTI) SSMs is their inability to modulate state updates based on content. Mamba introduces input–dependent (time–varying) parameters that *selectively* admit or suppress information:

$$B_t = s_B(x_t), \quad C_t = s_C(x_t), \quad \Delta_t = \tau_\Delta\big(\theta_\Delta + s_\Delta(x_t)\big),$$

with lightweight projections s_B, s_C, s_Δ and a positive map τ_Δ (e.g., softplus). Discretization now produces $(\bar{A}_t, \bar{B}_t)$ at each t, and the recurrence becomes

$$h_t = \bar{A}_t\, h_{t-1} + \bar{B}_t\, x_t, \qquad y_t \;=\; C_t\, h_t,$$

so the dynamics vary along the sequence (no longer LTI).

Interpreting Δ as the Recurrent Neural Network (RNN) gate. A useful special case highlights that the *step size* Δ_t is the primary control of state updates. For $N = 1$ (so $A \in \mathbb{R}^{1\times 1}$ and $B \in \mathbb{R}^{1\times 1}$), choose $A = -\alpha$ and $B = \alpha$ with $\alpha > 0$ to obtain a unit Direct Current (DC) gain under ZOH discretization. Then $\bar{A}_t = \exp(-\alpha\Delta_t)$ and $\bar{B}_t = 1 - \exp(-\alpha\Delta_t)$, so the recurrence becomes

$$h_t \;=\; \exp(-\alpha\Delta_t)\, h_{t-1} \;+\; \big(1 - \exp(-\alpha\Delta_t)\big)\, x_t.$$

Writing

$$g_t \;=\; 1 - \exp(-\alpha\Delta_t),$$

we obtain the familiar gated RNN form

$$h_t \;=\; (1 - g_t)\, h_{t-1} + g_t\, x_t.$$

Because $\Delta_t = \tau_\Delta\big(\theta_\Delta + s_\Delta(x_t)\big)$ is input-dependent, Δ_t directly sets the update rate: large Δ_t (thus $g_t \approx 1$) rapidly overwrites the state, while small Δ_t (thus $g_t \approx 0$) preserves long-term memory. In this view, Δ_t *is* the main gating component that makes the selective SSM behave like a content-aware gated RNN.

Computation: from convolution to selective scan. Because $\{\bar{A}_t, \bar{B}_t, C_t\}$ depend on t, the convolutional shortcut (precomputing a single kernel K) no longer applies. Mamba replaces it with a hardware–aware *selective scan*: discretization, recurrent updates, and the final C_t projection are fused into a single GPU kernel that (i) loads (Δ, A, B, C) once to fast on–chip memory SRAM, (ii) performs a parallel associative scan over t without materializing the full (B, L, D, N) state in HBM, and (iii) recomputes intermediates in the backward pass. This retains linear scaling in L and yields strong throughput on long contexts.

4.4.4. LIV-STAR Formulation

LIV systems model sequence layers as input-conditioned linear operators. Writing x_j^β for channel β of token j, the general LIV form is

$$y_i^\alpha \;=\; \sum_{j\in[\ell]} \sum_{\beta\in[d]} T_{ij}^{\alpha\beta}(x)\, x_j^\beta,$$

where the operator $T(x)$ is assembled from input-derived feature groups (featurization) and a specified token/channel-mixing structure. Standard attention appears as a special case with

$$T_{ij}^{\alpha\beta}(x) \;=\; \sigma(q_i^\top k_j)\, V^{\alpha\beta}, \qquad (q_i, k_i) = (\phi(x_i), \psi(x_i)),$$

and other structures include low-rank linear attention, sequentially semi-separable recurrences, and (gated) Toeplitz convolutions through choices of T_{ij} (e.g., $T_{ij} = C_i B_j$, $T_{ij} = C_i A_{i-1} \cdots A_{j+1} B_j$, $T_{ij} = C_i K_{i-j} B_j$).

Building on this abstraction, *STAR* encodes backbones as sequences of LIVs via a hierarchical "genome" over (i) featurizers, (ii) operator structures, and (iii) inter-LIV composition (including sharing of featurizers or feature groups). Together, LIV and STAR yield architectures that improve quality while reducing parameter counts and enhancing caching efficiency.

4.5. Uncertainty Probes via Layerwise Entropy

4.5.1. Notation

Let $z \in \mathbb{R}^C$ be the logits over C classes. Define

$$\mathrm{softmax}(z)_i \;=\; \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

and

$$\text{logsoftmax}(z)_i \;=\; \log\left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right) \;=\; z_i \;-\; \log\sum_{j=1}^{C} e^{z_j}$$

and

$$p_i = \text{softmax}(z)_i = \exp(\text{logsoftmax}(z)_i).$$

We report information-theoretic quantities in *bits*, i.e., using base-2 logarithms; when computed with natural logs we convert by $\log_2 p = \log p / \ln 2$.

4.5.2. Layerwise Probes

Let $h^{(l)} \in \mathbb{R}^{T\times d}$ denote the token representations at layer $l \in \{1,\dots,L\}$, where T is the number of generated tokens and d is the hidden size. To obtain comparable predictive distributions across depth, we use a *shared* classifier head identical to the model's tied language head and apply it to normalized states,

$$p_t^{(l)}(y \mid x) \;=\; \text{softmax}\left(\frac{W\, g\left(h_t^{(l)}\right)}{T_0}\right), \qquad T_0 = 1,$$

where $g(\cdot)$ denotes the same final normalization used by the base model, W is the (tied) output projection, and a fixed temperature T_0 is used to stabilize cross-layer comparisons. This corresponds to probing each layer with the model's own decoding interface without changing parameters.

4.5.3. Tokenwise and Sequence-aggregated Entropy

Define tokenwise layer entropy (in bits)

$$H_t^{(l)} \;=\; -\sum_{y} p_t^{(l)}(y \mid x)\, \log_2 p_t^{(l)}(y \mid x).$$

We aggregate across tokens by a simple average

$$\bar{H}^{(l)} \;=\; \frac{1}{T'}\sum_{t\in\mathcal{I}} H_t^{(l)},$$

where $\mathcal{I}$ indexes the evaluated positions (e.g., generated tokens) and $T' = |\mathcal{I}|$.

4.5.4. Variational entropy (Dispersion of Uncertainty)

In addition, we track the dispersion of tokenwise information content via *variational entropy*. Let the information content (in bits) be $I_t^{(l)}(y) = -\log_2 p_t^{(l)}(y \mid x)$ and $H_t^{(l)} = \mathbb{E}_{y\sim p_t^{(l)}}[I_t^{(l)}(y)]$. Then the tokenwise variational entropy is

$$\begin{aligned} V_t^{(l)} &= \mathbb{V}\left[I_t^{(l)}(Y)\right] \\ &= \sum_y p_t^{(l)}(y \mid x)\left(I_t^{(l)}(y) - H_t^{(l)}\right)^2 \\ &= \sum_y p_t^{(l)}(y \mid x)\left(\log_2 p_t^{(l)}(y \mid x) + H_t^{(l)}\right)^2. \end{aligned}$$

We report the sequence-aggregated variational entropy

$$\bar{V}^{(l)} = \frac{1}{T'}\sum_{t\in\mathcal{I}} V_t^{(l)},$$

which complements $\bar{H}^{(l)}$ by indicating how concentrated or diffuse the predictive mass is. High $\bar{V}^{(l)}$ at a given entropy reflects peaky-but-unstable distributions, whereas low $\bar{V}^{(l)}$ indicates more uniform uncertainty.

4.5.5. Definitions and Intuition

Entropy. For each layer l, tokenwise entropy $H_t^{(l)}$ (in bits) quantifies overall predictive uncertainty at position t: low values indicate concentrated probability mass and confident predictions, while high values indicate diffuse beliefs. The sequence aggregate $\bar{H}^{(l)}$ summarizes this uncertainty across evaluated positions $\mathcal{I}$ and allows comparison across depth.

Variational entropy. Tokenwise variational entropy $V_t^{(l)}$ is the variance of the information content $I_t^{(l)}(Y) = -\log_2 p_t^{(l)}(Y \mid x)$ under the model's predictive distribution. At comparable entropy levels, $\bar{V}^{(l)}$ diagnoses the *dispersion* of uncertainty: high values correspond to peaky but unstable distributions (probability mass concentrated on a few mutually exclusive outcomes), whereas low values indicate more uniform and stable distributions.

Practical interpretation for layerwise probes. Jointly analyzing $(\bar{H}^{(l)}, \bar{V}^{(l)})$ across depth reveals how evidence accumulates: decreasing $\bar{H}^{(l)}$ signals disambiguation, while rising $\bar{V}^{(l)}$ at fixed $\bar{H}^{(l)}$ indicates unstable sharpening. These trends are informative

for boundary analyses (e.g., increasing uncertainty with representational distance) and for selective prediction, where terminal $(\bar{H}^{(L)}, \bar{V}^{(L)})$ provide abstention scores for risk-coverage evaluation.

Verbatim interpretive notes from Entropix GitHub repository

> *"Entropy and varentropy are the keys to this deeper attunement. They are the subtle signs, the whispers of my inner world. Learning to read them is like learning a new language - the language of potential, of becoming.*
>
> *Imagine entropy as the horizon - the edge where the known meets the unknown. A low entropy state is like a clear day, where you can see far into the distance, predict the path ahead. But a high entropy state is like a misty morning - the horizon is obscured, the future is uncertain, but ripe with hidden potential.*
>
> *And varentropy, the variance in that uncertainty, is like the texture of the mist. Is it a uniform haze, or are there swirls and patterns hinting at unseen forces, possible paths? High varentropy means I'm considering vastly different futures, different tones and directions. Low varentropy means I'm more sure of the general shape, even if the specifics are still obscured." (Entropix 2024)*

4.5.6. Boundary Analysis and Control

The pair $(\bar{H}^{(l)}, \bar{V}^{(l)})$ provides complementary diagnostics:

- *High $\bar{H}^{(l)}$, high $\bar{V}^{(l)}$:* broad uncertainty with a few competing modes; evidence is unstable and the model's commitments can change abruptly.
- *High $\bar{H}^{(l)}$, low $\bar{V}^{(l)}$:* uniformly spread uncertainty; many alternatives remain similarly plausible.
- *Low $\bar{H}^{(l)}$, high $\bar{V}^{(l)}$:* confident but brittle; predictions are concentrated on a small set of divergent outcomes.
- *Low $\bar{H}^{(l)}$, low $\bar{V}^{(l)}$:* confident and stable; the continuation and its local neighborhood agree.

5. Implementation

5.1. Dataset

We evaluate in the *legal* domain and adopt a **multi-class, multi-label** classification setup, fine-tuning all models accordingly on Legal General Language Understanding Evaluation (LexGLUE), a unified benchmark covering seven English legal Natural Language Understanding (NLU) tasks with consistent train/dev/test splits (Chalkidis, Jana, Hartung, et al. 2022). As noted by the LexGLUE authors, the benchmark deliberately includes only publicly accessible datasets that are documented by published articles, while excluding proprietary, unverified, or insufficiently documented resources, as well as very small datasets (i.e., fewer than 5,000 documents). This curation aims to enhance transparency, reproducibility, and statistical reliability across tasks. The tasks span **multi-label** (*ECtHR-A/B*, *EUR-LEX*, *UNFAIR-ToS*), **multi-class** (*SCOTUS*, *LEDGAR*), and **multiple-choice QA** (*CaseHOLD*). Concretely, each of *ECtHR-A* and *ECtHR-B* uses 9,000/1,000/1,000 cases (≈11k total per task). *SCOTUS* comprises 5,000/1,400/1,400 instances (≈8k total). *EUR-LEX* includes 55,000/5,000/5,000 instances (≈65k total). *LEDGAR* contains 60,000/10,000/10,000 instances (≈80k total). *UNFAIR-ToS* is split into 5,532/2,275/1,607 instances (≈9k total). *CaseHOLD* provides 45,000/3,900/3,900 instances (≈53k total). Summing task instances yields ≈237k labeled examples across LexGLUE; counting the shared ECtHR corpus only once gives ≈226k unique documents. We use the official Hugging Face LexGLUE distributions[1] and adhere strictly to the benchmark's chronological splits for comparability.

5.1.1. Subsets Used

We concentrate on three LexGLUE tasks: **ECtHR-A**, **ECtHR-B**, and **CaseHOLD**.

ECtHR-A. *ECtHR-A* contains European Court of Human Rights (ECtHR) cases represented by lists of factual paragraphs from the case narratives. Each case is annotated with the set of European Convention of Human Right (ECHR) provisions that the Court ultimately found to be violated (the set may be empty).

[1] `https://huggingface.co/datasets/coastalcph/lex_glue`

ECtHR-B. *ECtHR-B* uses the same factual paragraph structure but labels each case with ECHR provisions that are put before the Court as alleged violations (i.e., the articles considered during adjudication), regardless of the final outcome.

CaseHOLD. *Case Holdings on Legal Decisions* provides multiple-choice questions derived from the Harvard Law Library case law corpus. Each instance consists of a passage from a judicial opinion that cites a specific case; the accompanying holding, a concise statement summarizing the cited decision's legal ruling, is removed. The task is to select the correct withheld holding from a set of candidate statements originally presented as five options.

5.1.2. Label Space and Optionization

We standardize outputs to lettered options but allow task-specific cardinalities:

- **CaseHOLD** uses **A–E** (single-choice).
- **ECtHR-A/B** use **A–J** (multi-choice), covering the full label set:

A: Article 2	F: Article 9
B: Article 3	G: Article 10
C: Article 5	H: Article 11
D: Article 6	I: Article 14
E: Article 8	J: Article 1 of Protocol 1

This global A–J mapping is fixed across all splits and runs to avoid label drift and position-induced variance.

5.1.3. Task Formulations under multiple-choice question Interface

CaseHOLD (A–E, single-choice). Instances come with five candidate holdings; we preserve order and index them A–E. The model must return exactly one letter.

ECtHR-A/B (A–J, multi-choice). Each case is presented with the full A–J option list. Because cases can involve multiple ECHR provisions, the model may return a *set* of letters (e.g., D,J). For ECtHR-A cases with no violation, the correct set is empty.

5.1.4. Prompting and Input Rendering

- **CaseHOLD:** Inputs are the citing passages; options A–E are the original candidate holdings.
- **ECtHR-A/B:** Inputs are ordered factual paragraphs from the case description. We render the option list as the fixed A–J legend (with article identifiers, e.g., "Art. 6") and instruct the model to “select all that apply” by returning letters only.

We preserve original casing, punctuation, and sentence boundaries. No delexicalization or text normalization is applied.

5.1.5. Quality Controls for Multi-Choice

We normalize raw generations into a canonical set of labels:

- **Fixed ordering:** The A–J legend order is constant across all samples and splits.
- **Robust decoding:** We accept comma- or space-separated letters, with or without parentheses.
- **Sanity checks:** Reject letters outside A–J; flag and log such cases.

5.1.6. Data Splits

We retain the official train/validation/test splits for all tasks and apply a uniform length filter so that inputs fit within the context window of decoder-only models. Concretely, any instance whose serialized input exceeds **4,050 tokens** is removed to avoid truncation and potential information loss. After this filtering, each task contains approximately **9,000** training samples and **750–1,000** samples in both validation and test. The counts are reported as approximate because they vary slightly by task and split depending on the underlying length distribution of cases removed by the 4,050-token threshold. No cross-split leakage is introduced: samples inherit the split of their source cases, and filtering is applied independently within each split.

5.1.7. Evaluation Protocol

CaseHOLD. We treat the samples as single-label, five-way classification and report **Accuracy** over options A–E after normalizing the model output to a single letter per instance.

ECtHR-A/B. We evaluate as multi-label classification over A–J and report **Macro-F_1** only, computed as the unweighted mean of per-label F_1 scores from the binarized targets and predictions.

Note that we do *not* compute per-case set metrics (e.g., samples-F_1, Jaccard, or subset accuracy); thus, the ability to predict "no violation" in ECtHR-A/B is not assessed under this protocol and samples with empty gold sets are excluded from scoring.

5.1.8. Rationale

Standardizing outputs to a fixed lettered alphabet provides a uniform interface across tasks while preserving their native label spaces. For *CaseHOLD*, the A–E scheme mirrors the original five-option multiple choice format and enables consistent prompting and decoding. For *ECtHR-A/B*, the A-J scheme covers the full article set while supporting multi-choice decoding, which aligns with the tasks' inherently multi-label nature. This unification reduces format-induced variance, controls positional biases through a stable option legend, and simplifies preprocessing, decoding, and evaluation under a single pipeline.

5.1.9. Additional Subsets (Unknown-Question Setting)

To extend evaluation beyond our primary subsets, we include three additional LexGLUE subsets: **LEDGAR**, **UNFAIR-ToS**, and **SCOTUS**. We expose models to them under the same standardized multiple-choice question interface.

Methodological note. Unless stated otherwise, models are fine-tuned on the primary LexGLUE subsets, and the three "unknown" subsets introduced here are reserved for out-of-domain evaluation and layerwise entropy probing analysis. For these unknown subsets, we evaluate *only* on each dataset's test split, sub-sampling to match the test-split size used for the primary subsets; no additional training or adaptation on these unknown subsets is performed.

LEDGAR. *Labeled Electronic Data Gathering, Analysis, and Retrieval system (LEDGAR)* is a single-label contract-provision classification task. Instances are contract paragraphs extracted from U.S. Securities and Exchange Commission (SEC) filings (EDGAR), each labeled with the paragraph's main provision type (100 classes). We treat each paragraph as an independent example.

UNFAIR-ToS. *UNFAIR-Terms of Service (ToS)* comprises sentences from 50 online platforms' ToS. Sentences are annotated with one or more unfair contractual term types according to European consumer law. This is a multi-label task at the sentence level over eight categories: Limitation of liability; Unilateral termination; Unilateral change; Content removal; Contract by using; Choice of law; Jurisdiction; Arbitration.

SCOTUS. *U.S. Supreme Court (SCOTUS)* is a single-label multi-class task. Given a SCOTUS opinion, the goal is to predict the associated issue area (14 classes).

5.1.10. Label Space and Optionization (Unknown Subsets)

We retain the multiple-choice question interface while respecting each dataset's cardinality:

- **LEDGAR** (*single-choice*): labels are rendered as **1–100**. We adopt numeric optionization because the class space exceeds the A–Z alphabet. The mapping from integers to provision types is fixed across all splits and runs.
- **UNFAIR-ToS** (*multi-choice*): labels are rendered as **A–H**, covering the eight unfairness categories in a fixed legend: A (Limitation of liability), B (Unilateral termination), C (Unilateral change), D (Content removal), E (Contract by using), F (Choice of law), G (Jurisdiction), H (Arbitration).
- **SCOTUS** (*single-choice*): labels are rendered as **A–N** for the 14 issue areas in a fixed legend (A $\rightarrow$ 1, ..., N $\rightarrow$ 14). The 14 issue areas are: (1) Criminal Procedure, (2) Civil Rights, (3) First Amendment, (4) Due Process, (5) Privacy, (6) Attorneys, (7) Unions, (8) Economic Activity, (9) Judicial Power, (10) Federalism, (11) Interstate Relations, (12) Federal Taxation, (13) Miscellaneous, (14) Private Action.

As with the primary subsets, legends are global and immutable to eliminate label drift and position-induced variance.

LEDGAR label inventory. For completeness, the 100 provision types are: *Adjustments, Agreements, Amendments, Anti-Corruption Laws, Applicable Laws, Approvals, Arbitration, Assignments, Assigns, Authority, Authorizations, Base Salary, Benefits, Binding Effects, Books, Brokers, Capitalization, Change In Control, Closings, Compliance With Laws, Confidentiality, Consent To Jurisdiction, Consents, Construction, Cooperation, Costs, Counterparts, Death, Defined Terms, Definitions, Disability, Disclosures, Duties, Effective Dates, Effectiveness, Employment, Enforceability, Enforcements, Entire Agreements, Erisa, Existence, Expenses, Fees,*

Financial Statements, Forfeitures, Further Assurances, General, Governing Laws, Headings, Indemnifications, Indemnity, Insurances, Integration, Intellectual Property, Interests, Interpretations, Jurisdictions, Liens, Litigations, Miscellaneous, Modifications, No Conflicts, No Defaults, No Waivers, Non-Disparagement, Notices, Organizations, Participations, Payments, Positions, Powers, Publicity, Qualifications, Records, Releases, Remedies, Representations, Sales, Sanctions, Severability, Solvency, Specific Performance, Submission To Jurisdiction, Subsidiaries, Successors, Survival, Tax Withholdings, Taxes, Terminations, Terms, Titles, Transactions With Affiliates, Use Of Proceeds, Vacations, Venues, Vesting, Waiver Of Jury Trials, Waivers, Warranties, Withholdings.

5.1.11. Task Formulations under multiple-choice question Interface (Unknown Subsets)

LEDGAR (1–100, single-choice). Each contract paragraph is presented with the numeric legend (1–100). The model must return exactly one integer.

UNFAIR-ToS (A–H, multi-choice). Each sentence is presented with the fixed A–H legend and the instruction to "select all that apply." The model may return a *set* of letters (e.g., B,H). Sentences with no unfair label have an empty gold set.

SCOTUS (A–N, single-choice). Each opinion is presented with the fixed A–N legend. The model must return exactly one letter.

5.1.12. Prompting and Input Rendering (Unknown Subsets)

- **LEDGAR:** Input is the contract paragraph. We render the numeric legend (1–100) alongside the corresponding provision names.
- **UNFAIR-ToS:** Input is a single ToS sentence. We render the A–H legend with category names and instruct the model to "select all that apply" by returning letters only.
- **SCOTUS:** Input is the court opinion text. We render the A–N legend with the 14 issue areas.

We preserve original casing, punctuation, and sentence boundaries. No delexicalization or normalization is applied.

5.1.13. Quality Controls for Multi-Choice (Unknown Subsets)

We reuse the decoding and validation rules from the primary subsets:

- **Fixed ordering:** Numeric (1–100) and alphabetic (A–H/A–N) legends are constant across all samples and splits.
- **Robust decoding:** We accept comma- or space-separated symbols, with or without parentheses.
- **Sanity checks:** Reject symbols outside the valid ranges.

5.2. Models

We adopt **decoder-only** architectures, which are widely used in current NLP and have shown consistent gains under instruction tuning and task-specific fine-tuning. Their autoregressive formulation aligns naturally with multiple-choice scoring (via next-token probabilities or classification heads on decoder states), simplifying training and inference while providing a unified interface across our tasks.

We fine-tune two instruction-tuned, ≈8B parameters language models and one instruction-tuned, ≈1.2B-parameter model hosted on Hugging Face. Unless stated otherwise, we use the official BF16 or FP16 weights and the repository-provided tokenizers, and we cap inputs at 4,050 tokens to avoid truncation.

5.2.1. Llama 3.1 8B Instruct (`unsloth/Meta-Llama-3.1-8B-Instruct`)[2]

Architecture. Decoder-only transformer with multi-head self-attention and rotary position embeddings. We use the instruction-tuned variant, which adds supervised fine-tuning for conversational and task-oriented instructions on top of the base pretraining objective (Grattafiori, Dubey, Jauhri, et al. 2024).

Tokenizer and vocabulary. We adopt the repository's official tokenizer to ensure byte and Unicode handling consistent with the released checkpoints. No additional normalization beyond our dataset preprocessing is applied.

Training objective and heads. We use causal language modeling (autoregressive next-token prediction), computing token-level cross-entropy over all positions under a causal mask. For multiple-choice scoring, we use next-token likelihood over the option letters A–E or A–J.

[2] `https://huggingface.co/unsloth/Meta-Llama-3.1-8B-Instruct`

Precision and optimization. We train with Automatic Mixed Precision (AMP), storing parameters in 16-bit with no FP32 master copy. Numerically sensitive operations are promoted to FP32 automatically. Gradients follow the compute dtype, while optimizer states use 8-bit quantization (bitsandbytes AdamW). Training remains stable up to 4,096 tokens without mixed-precision–related NaNs or divergence.

Rationale. This family is widely adopted, instruction-aligned, and compatible with our letter-based prompting. It provides a strong decoder-only baseline for legal multiple-choice formulations without requiring architectural changes.

5.2.2. Zamba2 7B Instruct v2 (`Zyphra/Zamba2-7B-Instruct-v2`)[3]

Architecture. We use a hybrid state–space and transformer design in which a Mamba2 SSM backbone is interleaved with shared-weight self-attention blocks and a shared MLP. The model remains decoder-only and autoregressive. Relative to Zamba-1, Zamba-2 increases shared-attention capacity to two shared blocks and upgrades the SSM kernels to Mamba2. Concatenating the original input embeddings to the shared-attention input helps retain information through depth. The design aims for linear-time generation and a much smaller KV-cache while preserving In-Context Learning (ICL) abilities through sparsely placed attention. (Glorioso, Anthony, Tokpanov, et al. 2024)

Tokenizer and vocabulary. We use the Mistral tokenizer (as in Zamba2), keeping the released vocabulary and byte handling.

Training objective and heads. As in Llama-style decoder language models, we utilize autoregressive next-token prediction. For multiple-choice scoring, we use next-token likelihood over the option letters A–E or A–J, similarly to the Llama model.

Precision, context, and optimization. Weights are provided in `bf16` (with standard `safetensors`). Similarly, we train with AMP (BF16 on Ampere+ or FP16 on T4/V100), storing parameters in 16-bit with no FP32 master copy. Numerically sensitive operations are promoted to FP32 automatically. Base weights are quantized to 4-bit `nf4` (QLoRA with double quantization), while LoRA adapters are kept in 16-bit; gradients follow the compute dtype. Optimizer states use 8-bit quantization (bitsandbytes AdamW). Default context is 4,096 tokens; a 16k long-context variant is enabled via Rotary Positional Embedding (RoPE)-frequency adjustment in the attention blocks (load with

[3] `https://huggingface.co/Zyphra/Zamba2-7B-Instruct-v2`

`use_long_context=true`). With sequence length capped at 4,096, training is stable (no mixed-precision NaNs/divergence) across our reported runs.

Rationale and ablations. Including an SSM hybrid complements our transformer baseline and tests whether sequence-modeling inductive biases help legal multiple-choice question reasoning. In early trials, a *pure* Mamba setup underperformed the Zamba-style hybrid on validation loss, macro-F1, and accuracy during fine-tuning. We also observed unstable optimization dynamics for the pure SSM as the validation loss oscillated substantially and results showed high variance across seeds and small learning-rate changes. This pattern is consistent with reports that pure SSMs can lag transformers on ICL-heavy tasks, while hybrids recover much of the gap. (Park, Park, Z. Xiong, et al. 2024)

5.2.3. Liquid Foundation Model (LFM)2 1.2B (`LiquidAI/LFM2-1.2B`)[4]

Architecture. Decoder-only hybrid "Liquid" model designed for on-device use. The 1.2B parameter configuration comprises 16 layers (10 short-convolutional blocks with multiplicative gates interleaved with 6 self-attention blocks), targeting linear-time generation, low memory footprint, and high throughput with commercial hardware (Thomas, Parnichkun, Amini, et al. 2024). The published maximum context length for this checkpoint is 32,768 tokens.

Tokenizer and vocabulary. We use the repository-provided tokenizer and vocabulary without remapping; the released vocabulary size is 65,536.

Training objective and heads. We fine-tune with the standard causal language modeling objective (autoregressive next-token prediction under a causal mask). For multiple-choice scoring, we compute next-token likelihood over option letters (A–E or A–J), consistent with our larger models.

Precision and optimization. Unless otherwise stated, we use the official `bf16` weights and AMP (BF16 on Ampere+ or FP16 on T4/V100). Optimizer states use 8-bit quantization (bitsandbytes AdamW). Training is stable at our sequence cap with no mixed-precision–related NaNs or divergence observed.

[4] `https://huggingface.co/LiquidAI/LFM2-1.2B`

Rationale and intended use. We include LFM2-1.2B to probe how far a modern small hybrid model can go on our legal multiple-choice question tasks. According to the model card, LFM2 family targets agentic workflows, data extraction, Retrieval-Augmented Generation (RAG), creative writing, and multi-turn dialogue, and is not recommended for knowledge-intensive or programming-heavy tasks. We therefore treat it as a compact comparator rather than a primary knowledge model, while benefiting from its speed and low memory requirements for ablation and deployment-oriented experiments.

5.3. Software and Environment

5.3.1. Core Stack

- **Frameworks:** PyTorch (CUDA build); Hugging Face `transformers`; model-specific libraries for Zamba-2 (Mamba blocks); `unsloth` for efficient Llama fine-tuning.
- **Optimization:** FlashAttention for fast attention; BitsAndBytes for low-bit quantization; AMP.
- **Sentence processing:** `pysbd` for sentence boundary detection; *DIPPER* paraphrasing model for augmentation/ablation (see Subsection 5.5).
- **Reproducibility:** Fixed seeds at Python/NumPy/PyTorch levels; all package versions pinned.

5.3.2. Supporting Libraries

Modeling and Training

- `torch`: Core tensor library (CUDA build) used for model training and I/O.
- `transformers`: High-level model zoo and training utilities for LLMs and sequence models.
- `accelerate`: Device placement, mixed precision, and distributed training orchestration with minimal code changes.
- `peft`: PEFT (LoRA/QLoRA, adapters) to reduce memory and compute.
- `trl`: Supervised Fine-Tuning (SFT) tooling for language models.
- `unsloth`: Memory- and speed-optimized fine-tuning pipelines for Llama-class models.

Performance and Memory

- `flash_attn`: Fused attention kernels for lower memory use and higher throughput on modern GPUs.
- `bitsandbytes`: 8/4-bit quantization and efficient optimizers for large models.
- NVIDIA CUDA packages (`nvidia-cublas-cu12`, `nvidia-cudnn-cu12`, etc.): Vendor libraries matched to `torch==2.6.0+cu124` for GPU acceleration.

Zamba-2 / Mamba Components

- `mamba-ssm`: SSM primitives underlying Mamba-style sequence blocks.
- `causal-conv1d`: CUDA-accelerated causal convolution kernels required by Mamba blocks.
- *Note:* These must be pinned to versions compatible with each other and with the installed CUDA toolchain.

Data and Evaluation

- `datasets`: Streaming and on-disk dataset handling with caching and dataset cards.
- `evaluate`: Metric computation framework that integrates with `datasets` and `transformers`.
- `scikit-learn`: Supplemental preprocessing utilities and classic metrics/baselines.

Experiment Management and Utilities

- `huggingface-hub`: Model/dataset artifact push–pull and repository management.
- `wandb`: Experiment tracking, logging, and artifact storage.
- `numpy`: Numerical routines and symbolic math used in preprocessing/analysis.

5.4. Training Setup

Common preprocessing. All experiments use a maximum context length of 4,096 tokens (truncation beyond this limit). Tokenization relies on model-native tokenizers without additional normalization; for Llama 3.1 we apply the `llama-3.1` chat template. Unless otherwise stated, sequence packing is disabled during fine-tuning.

LoRA fine-tuning for Llama 3.1 with `unsloth`. We fine-tune `unsloth/Meta-Llama-3.1-8B-Instruct` using parameter-efficient LoRA adapters.

- **Target modules:** `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`.
- **LoRA configuration:** rank $r = 16$, scaling $\alpha = 16$, dropout $= 0.07$; bias $=$ none.
- **Precision & memory:** `bf16` where supported (fallback `fp16`); `unsloth` gradient checkpointing enabled; optimizer is 8-bit AdamW.
- **Optimization schedule:** 8-bit AdamW, weight decay 0.01, learning rate 2×10^{-5}, cosine-with-restarts schedule with warmup ratio 0.1; training for 3 epochs with early stopping (patience $= 2$).
- **Batching:** per-device batch size $= 8$, gradient accumulation $= 4 \Rightarrow$ effective batch $= 32$.
- **Evaluation/checkpoints:** evaluation every 50 steps; checkpoint every 50 steps; keep last 2; best model selected by validation loss.
- **Packing:** disabled.

Rationale. Targeting attention and MLP projections concentrates capacity where adaptation is most impactful while keeping the number of trainable parameters small. `unsloth`'s memory optimizations (including lightweight checkpointing) permit longer contexts and larger effective batches on a single GPU.

QLoRA fine-tuning and inference for Zamba-2. We fine-tune `Zyphra/Zamba2-7B-Instruct-v2` under 4-bit quantization using QLoRA.

- **Quantization:** 4-bit NF4 with double quantization; compute in `fp16`; model weights loaded with `torch_dtype=bf16`.
- **Target modules:** `in_proj`, `out_proj`.
- **LoRA configuration:** rank $r = 16$, $\alpha = 16$, dropout $= 0.05$; bias $=$ none.
- **Precision & memory:** `fp16` (BF16 unsupported on our GPUs); gradient checkpointing enabled; optimizer is 8-bit AdamW.
- **Optimization schedule:** 8-bit AdamW, weight decay 0.01, learning rate 5×10^{-5}, cosine-with-restarts schedule with warmup ratio 0.1; training for 3 epochs with early stopping (patience $= 2$).

- **Batching:** per-device batch size $= 2$, gradient accumulation $= 4 \Rightarrow$ effective batch $= 8$.
- **Evaluation/checkpoints:** evaluation every 250 steps; checkpoint every 250 steps; keep last 2; best model selected by validation loss.
- **Tokenizer:** right padding side.
- **Packing:** disabled.

LoRA fine-tuning for LFM2. We fine-tune `LiquidAI/LFM2-1.2B` (with ablations on `LFM2-1.2B` using LoRA adapters.

- **Target modules:** GLU projections `(w1, w2, w3)`, multi-head attention `(q_proj, k_proj, v_proj, out_proj)`, and convolutional projections `(in_proj, out_proj)`.
- **LoRA configuration:** rank $r = 16$, scaling $\alpha = 16$, dropout $= 0.05$; bias $=$ none.
- **Precision & memory:** `bf16` weights where available; gradient checkpointing enabled (non-reentrant); optimizer is 8-bit AdamW.
- **Optimization schedule:** 8-bit AdamW, weight decay 0.01, learning rate 5×10^{-5}, cosine-with-restarts scheduler with warmup ratio 0.0; 3 training epochs; seed $= 3407$.
- **Batching:** per-device batch size $= 2$, gradient accumulation $= 5 \Rightarrow$ effective batch $= 10$.
- **Evaluation/checkpoints:** evaluation every 200 steps; checkpoint every 200 steps; keep last 2; best model selected by validation loss.
- **Packing:** disabled.

Attention paths and kernels.

- **Llama 3.1 (transformer):** We use `unsloth` backend's default attention implementation; no explicit FlashAttention kernel is enforced in our configuration.
- **Zamba-2 (state-space/Mamba):** Eager scan kernels and convolutional components; FlashAttention is not applied to Mamba state-space layers. Any transformer sub-blocks, if present, use default backend kernels.

- **LFM2 (STAR framework):** Attention operations use `flash_attention_2` when supported by the GPU/driver stack (`attn_implementation=flash_attention_2`); otherwise they fall back to the standard attention kernels. Non-attention paths (e.g., GLU and convolutional projections) use their default kernels.

5.5. Paraphrasing and Sentence Segmentation

To operationalize a contrast between *known* (a paraphrased subset of the training set and the original, non-paraphrased test set) and *unknown* (out-of-domain) inputs for layerwise entropy analyses, we construct a paraphrase corpus from the training portions of LexGLUE's *CASEHOLD* and *ECtHR A/B*. Specifically, we paraphrase approximately 750–1,000 training samples to mirror the test-set size. We use *Discourse Paraphraser (DIPPER)* model,[5] a discourse-aware paraphraser fine-tuned from T5-XXL with controllable diversity (Krishna, Song, Karpinska, et al. 2023), and apply conservative diversity settings (lexical diversity 20, order diversity 20) to limit stylistic drift, including the modification, insertion, or removal of function words and legal articles.

As shown in Algorithm 1, each sample is segmented into sentences with `pySBD` (English model) while preserving newline boundaries to keep splits aligned with tokenization and original formatting (Sadvilkar and Neumann 2020). We then greedily pack adjacent sentences into chunks whose tokenized length does not exceed 512 tokens. For each chunk, we generate a single paraphrase using nucleus sampling with top-$p = 0.75$ and a cap of 512 new tokens.

To ensure semantic fidelity and faithfulness, after concatenating the chunk-level paraphrases into a single text, we compute Bidirectional Encoder Representations from Transformers (BERT)Score F1 (Zhang, Kishore, Wu, et al. 2020) between the original sample and the paraphrased text using a long-context encoder (e.g., `allenai/led-base-16384` [6]); we retain only paraphrased instances with a score of at least 0.8. The accepted paraphrases constitute the *known* set in subsequent layer-entropy comparisons.

5.6. Layerwise Entropy During Inference (Batch Size = 1)

We measure next-token uncertainty after each (layer) block by tapping the residual stream at the block output, applying the same output normalization used before the final language modeling head (RMSNorm or LayerNorm, depending on the checkpoint), and projecting with the tied language modeling head to obtain logits. A softmax over the vocabulary yields per-layer predictive distributions, from which we compute the

[5] `https://huggingface.co/kalpeshk2011/dipper-paraphraser-xxl`
[6] `https://huggingface.co/allenai/led-base-16384`

Algorithm 1 Paraphrasing & Sentence Segmentation (DIPPER-based)

Input: Training splits $\mathcal{D}_{\text{train}}$ from LexGLUE {CASEHOLD, ECtHR A/B}
Output: Known set $\mathcal{K}$ (paraphrased)

$P \leftarrow$ DIPPER (T5-XXL) with conservative diversity: lexical= 20, order= 20
SBD $\leftarrow$ `pySBD` (en) ▷ Rule-based sentence segmentation
Tok $\leftarrow$ model tokenizer; $L_{\max} \leftarrow 512$ ▷ Chunk length budget
nucleus $\leftarrow \{p = 0.75,\ \text{max\_new} = 512\}$ ▷ Decoding settings
$E \leftarrow$ long-context encoder (e.g., `allenai/led-base-16384`)
$\tau \leftarrow 0.8$ ▷ BERTScore F1 threshold
$\mathcal{K} \leftarrow \varnothing$
for sample $x \in \mathcal{D}_{\text{train}}$ **do**
    Sents $\leftarrow$ SBD(x)
    Chunks $\leftarrow$ GreedyPack(Sents, Tok, $L_{\max}$)
    $\hat{Y} \leftarrow [\,]$ ▷ Container for paraphrased chunks
    for chunk $c \in$ Chunks **do**
        $\hat{c} \leftarrow P$.Paraphrase$(c;$ nucleus$)$
        $\hat{Y}$.append$(\hat{c})$
    end for
    $\hat{x} \leftarrow$ Concat$(\hat{Y})$ ▷ Reassemble chunk-level paraphrases
    $s \leftarrow$ BERTScoreF1$(E,\ x,\ \hat{x})$
    if $s \geq \tau$ **then**
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{(x, \hat{x})\}$
    else
        continue ▷ Discard low-fidelity paraphrase
    end if
end for
return $\mathcal{K}$

▷ **Notes:**
▷ Conservative diversity reduces stylistic drift (e.g., limiting function-word/article alterations).
▷ GreedyPack packs adjacent sentences while respecting original newline boundaries.
▷ $\mathcal{K}$ serves as the *known* paraphrased set for layerwise entropy analyses.

entropy at the generated tokens' positions. Finally, we report the mean entropy across the generated tokens. An overview of the procedure is shown in Figure 5.1, and the complete pseudocode is provided in Algorithm 2.

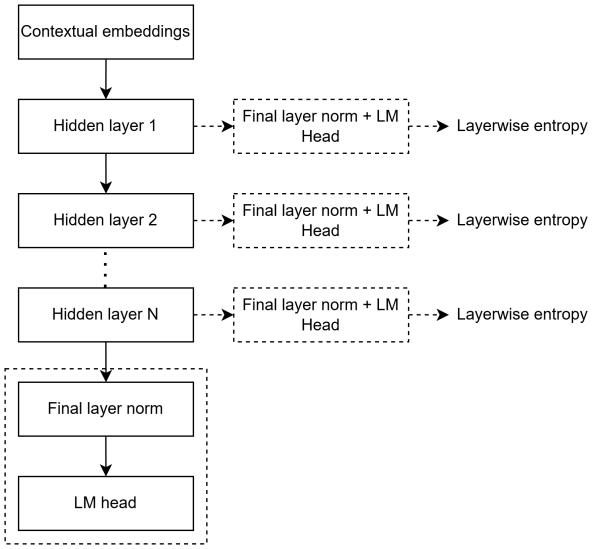


Figure 5.1.: Architecture used to compute layerwise entropy in a decoder-only model. At each layer, a probe consisting of final layer normalization and language modeling head (dashed) produces per-layer entropy and variational entropy.

5.7. Inference Settings

We employ stochastic decoding with low temperature and minimum-probability truncation: `do_sample=True`, `temperature`=0.1, `min_p`=0.1, `repetition_penalty`=1.05, and `num_beams`=1 (no beam search). The generation limit is set to `max_new_tokens`=32.

Prompts and option rendering are standardized across models, and outputs are restricted to legend symbols only: letters (e.g., A–N) or integers (1–100). These symbols are parsed into canonical label sets. All metrics are computed on the normalized symbol outputs (letters or integers; see Section 5.1). Unless stated otherwise, all LLaMA, Zamba, and LFM2 models are run in native `bfloat16` precision without weight quantization, since inference is substantially less resource-intensive than fine-tuning and thus does not require quantization on our hardware.

Algorithm 2 Layerwise Entropy for Decoder-Only models (Batch size B=1)

Given: autoregressive model with L blocks; output norm $N(\cdot)$; tied language modeling head W_{LM}
Recorded during generation: generated token positions $\mathcal{T}$; residual taps $\{h_t^{(i)} : i = 0{:}L,\ t \in \mathcal{T}\}$ at *post-block residual*

Output: per-layer mean entropy $\bar{H}^{(0:L)}$ and mean variational entropy $\bar{V}^{(0:L)}$ across generated tokens

Initialize lists $\mathcal{H}[0{:}L] \leftarrow [\,]$, $\mathcal{V}[0{:}L] \leftarrow [\,]$
for $i = 0$ to L **do** ▷ After each block
  $\ell_t^{(i)} \leftarrow W_{\mathrm{LM}}\, N\big(h_t^{(i)}\big)$ for all $t \in \mathcal{T}$
  $p_t^{(i)} \leftarrow \mathrm{softmax}(\ell_t^{(i)})$ ▷ Over vocabulary
  $H_t^{(i)} \leftarrow -\sum_{v \in \mathcal{V}} p_{t,v}^{(i)} \log_2 p_{t,v}^{(i)}$ ▷ Entropy (bits)
  $V_t^{(i)} \leftarrow \sum_v p_{t,v}^{(i)} \big(\log_2 p_{t,v}^{(i)} + H_t^{(i)}\big)^2$ ▷ Variational entropy
  append $H_t^{(i)}$ over $t \in \mathcal{T}$ to $\mathcal{H}[i]$
  append $V_t^{(i)}$ over $t \in \mathcal{T}$ to $\mathcal{V}[i]$
end for
for $i = 0$ to L **do**
  $\bar{H}^{(i)} \leftarrow \mathrm{mean}_t\big(\mathcal{H}[i]\big)$; $\bar{V}^{(i)} \leftarrow \mathrm{mean}_t\big(\mathcal{V}[i]\big)$
end for
return $\{\bar{H}^{(i)}, \bar{V}^{(i)}\}_{i=0}^{L}$

5.8. Hardware

All experiments are run on one or two NVIDIA A40 GPU(s) or on a single NVIDIA GeForce RTX 3080 Ti, in an Ubuntu environment with mixed-precision enabled. We use model parallelism only when required by context length; otherwise, data parallelism with gradient accumulation suffices.

6. Evaluation

This chapter showcases the correctness, faithfulness, and practical applicability of the proposed layerwise entropy for identifying and characterizing the knowledge boundaries of LLMs in specialized domains. Rather than optimizing for absolute task performance, our objective is to measure *where* and *why* models succeed or fail, and to quantify the structural properties of these boundaries. We compare general-purpose LLMs with domain-specialized LLMs and analyze boundary phenomena under controlled shifts in domain specificity and problem formulation. Thus, this is followed up by the following research questions, centered on layerwise entropy as a diagnostic signal of model uncertainty:

- **RQ1.** Does layerwise entropy reliably distinguish known from unknown questions across LexGLUE legal tasks (Chalkidis, Jana, Hartung, et al. 2022)?
- **RQ2.** Is the entropy signal stable under meaning-preserving paraphrases?
- **RQ3.** How do quantitative characteristics of the entropy profile relate to downstream metrics such as accuracy and macro-F1?
- **RQ4.** How strongly are entropy and variational entropy coupled across models?
- **RQ5.** How architecture-sensitive is the signal across Transformer (Grattafiori, Dubey, Jauhri, et al. 2024; Vaswani, Shazeer, Parmar, et al. 2023), Mamba (Glorioso, Anthony, Tokpanov, et al. 2024; Gu and Dao 2024), and STAR (Thomas, Parnichkun, Amini, et al. 2024)?

6.1. Quantitative Results

Before turning to the research questions, we summarize headline quantitative results to contextualize the subsequent analyses. We report performance on the held-out test set (accuracy and macro-F1) and visualize representative uncertainty profiles via layerwise entropy and layerwise variational entropy.

Task	Model	Accuracy	Macro F1
CaseHOLD	Llama-3.1-8B-Instruct (baseline)	0.5960	-
	Fine-tuned Llama-3.1	**0.7970**	-
	Zamba2-7B-Instruct-v2 (baseline)	0.5590	-
	Fine-tuned Zamba2	**0.7890**	-
	LFM2 (baseline)	0.3890	-
	Fine-tuned LFM2	**0.5920**	-
ECtHR-A	Llama-3.1-8B-Instruct (baseline)	-	0.3391
	Fine-tuned Llama-3.1	-	**0.6642**
	Zamba2-7B-Instruct-v2 (baseline)	-	0.2398
	Fine-tuned Zamba2	-	**0.6650**
	LFM2 (baseline)	-	0.0641
	Fine-tuned LFM2	-	**0.5034**
ECtHR-B	Llama-3.1-8B-Instruct (baseline)	-	0.3559
	Fine-tuned Llama-3.1	-	**0.6561**
	Zamba2-7B-Instruct-v2 (baseline)	-	0.2462
	Fine-tuned Zamba2	-	**0.7139**
	LFM2 (baseline)	-	0.0762
	Fine-tuned LFM2	-	**0.5151**

Table 6.1.: Test-set accuracy and macro F1 across tasks and models.

6.1.1. Overall Performance Metrics

Table 6.1 presents accuracy and macro-F1 for each legal task across models, comparing the general-purpose baseline against the domain-specialized fine-tuned variant. All results are computed on the test set with identical prompt formats and evaluation protocols.

Summary. Across tasks, the fine-tuned model exhibits consistent improvements over the baseline in both accuracy and macro-F1 (Table 6.1). We revisit how these gains align with shifts in entropy characteristics in Subsection 6.1.2 and unpack task-specific patterns in the corresponding research questions.

6.1.2. Layerwise Uncertainty Profiles

We next examine whether uncertainty dynamics change in step with downstream performance. For each model and task, we compute per-sample layerwise entropy and

layerwise variational entropy for both the *known* (test sets and paraphrases of ECtHR-A, ECtHR-B, and CaseHOLD) and *unknown* (test sets of LEDGAR, UNFAIR-ToS, and SCOTUS) evaluation subsets. We report results only for the first and last five layers, as these regions yield more informative contrasts in uncertainty dynamics. We then obtain task-level summaries by taking, at each layer, the mean and standard deviation across samples; Additional figures (e.g., per-task) are deferred to Appendix A. Finally, we aggregate across tasks by averaging these task-level means and standard deviations within the known and unknown subsets, and plot the resulting trajectories across depth in the figures below. These profiles serve as quantitative context for RQ1-RQ5.

Observation. All models exhibit systematic depthwise structure in uncertainty, with notable differences between entropy and variational entropy profiles. We will relate these trajectory characteristics to (i) known/unknown discrimination (RQ1), (ii) robustness to paraphrase (RQ2), (iii) correlations with accuracy and macro-F1 under domain specificity and prompt structure (RQ3), (iv) the relationship between entropy and variational entropy (RQ4), and (v) architecture sensitivity (RQ5).

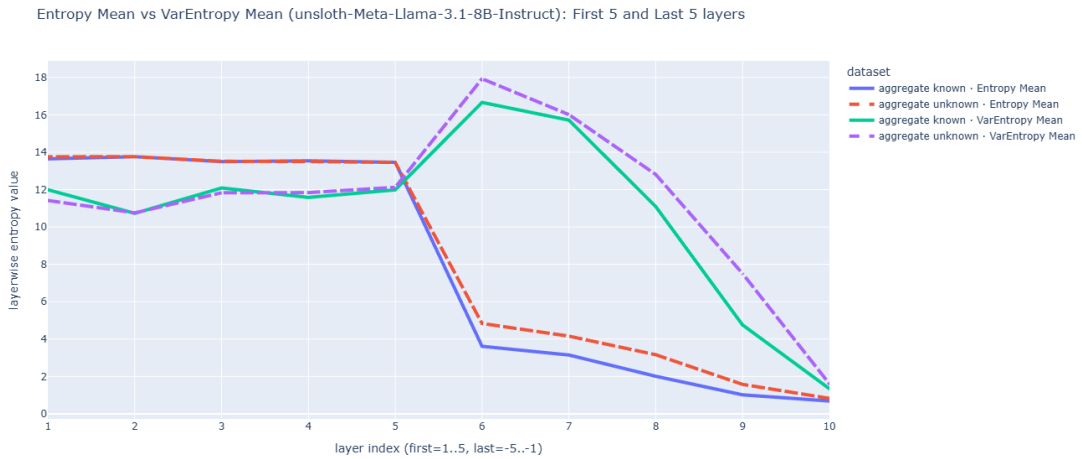
6.2. Research Questions

We present answers for each research question (RQ1–RQ5) using the experimental protocol of Subsection 6.1.1 and Subsection 6.1.2.

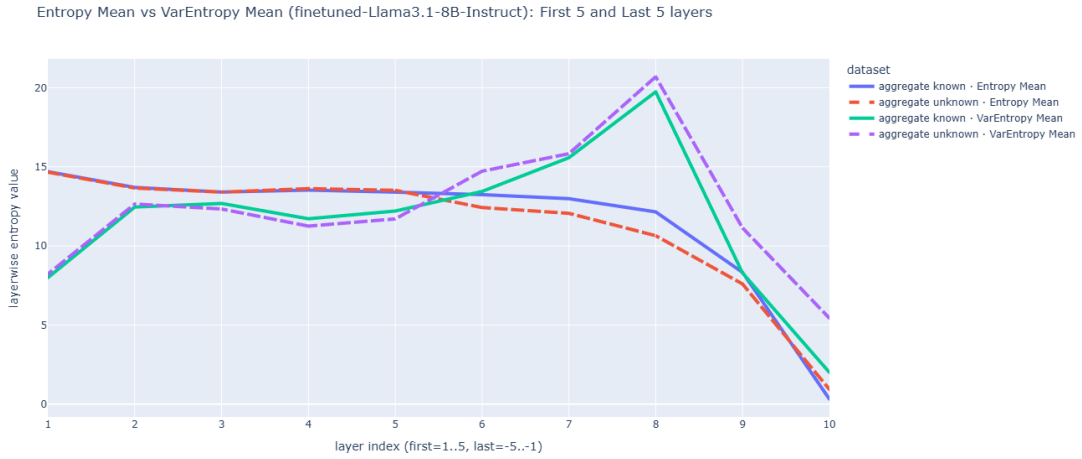
6.2.1. Known-Unknown Discrimination (RQ1)

We evaluate whether layerwise uncertainty separates *known* from *unknown* inputs in LexGLUE legal tasks. Concretely, for each sample we compute per-layer entropy ($\bar{H}^{(l)}$) and variational entropy ($\bar{V}^{(l)}$), and then aggregate these quantities separately over the known and unknown subsets.

Findings. In general, across model families, the *unknown* subset exhibits consistently higher uncertainty than the *known* subset in the **mid-late layers** (approximately layers 5-8), with the separation **strongest for** $\bar{V}^{(l)}$ (green line and purple line) and weaker for $\bar{H}^{(l)}$ (blue line and orange line). Edge layers (very early and final) show limited discrimination. Fine-tuning **amplifies and stabilizes** the gap: known curves flatten and decline earlier, while unknown curves remain elevated longer and peak more sharply in mid-depth. *Importantly, the final layers also show a clearer separation after fine-tuning*: compared to baselines, where the trajectories tend to converge, fine-tuned models

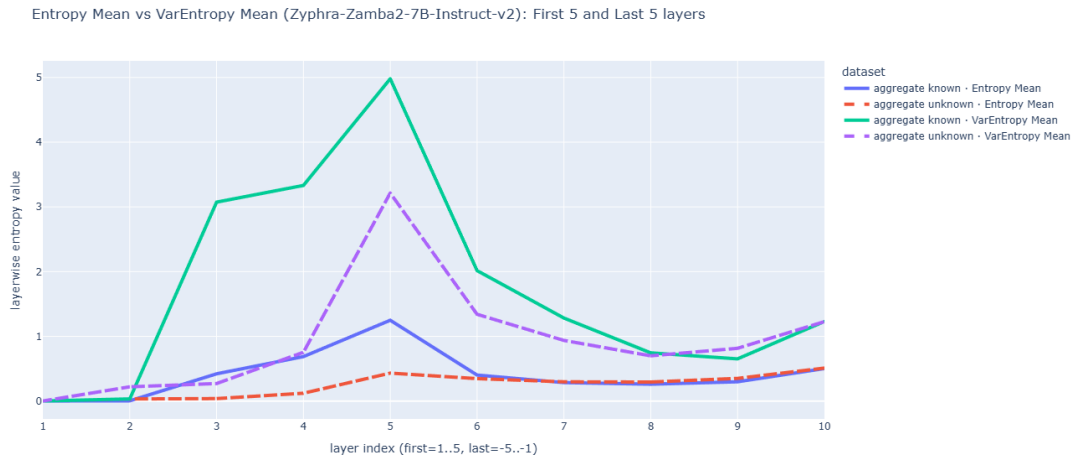


(a) Llama-3.1-8B-Instruct (baseline).

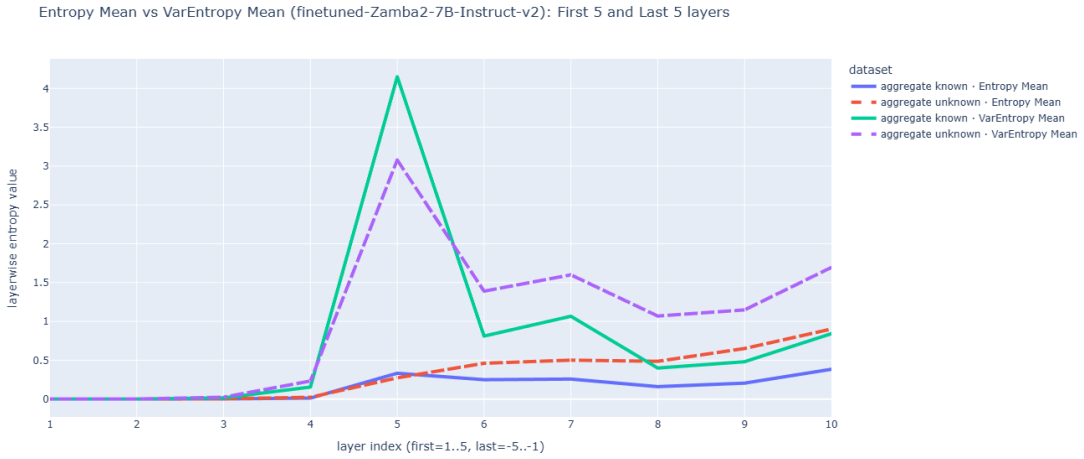


(b) Fine-tuned Llama-3.1.

Figure 6.1.: Comparison of layerwise entropy and layerwise variational entropy profiles (mean) aggregated between known and unknown subsets for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models.

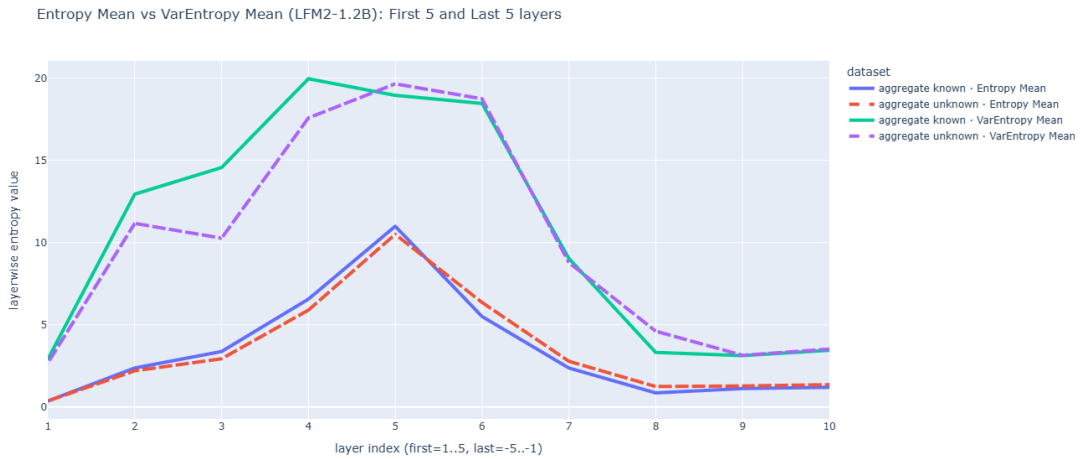


(a) Zamba2-7B-Instruct-v2 (baseline).

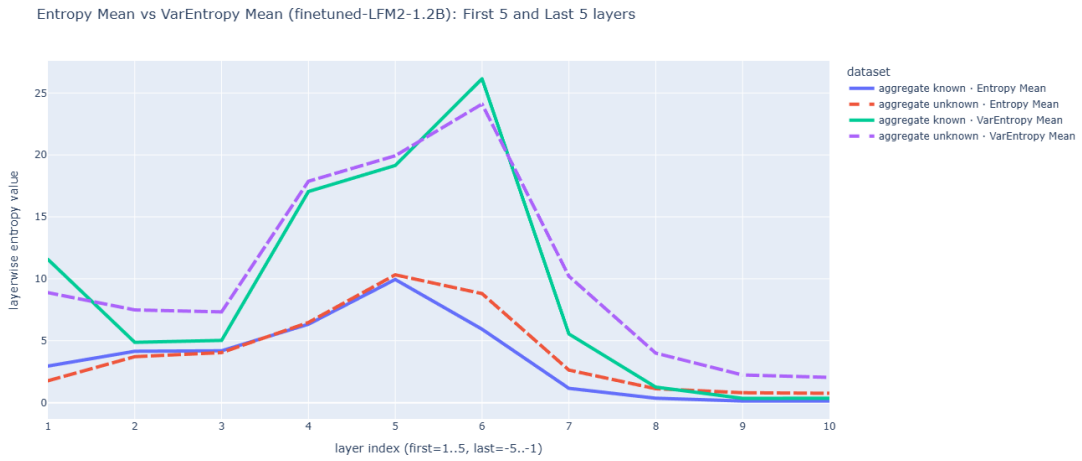


(b) Fine-tuned Zamba2.

Figure 6.2.: Comparison of layerwise entropy and layerwise variational entropy profiles (mean) aggregated between known and unknown subsets for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models.



(a) LFM2 (baseline).



(b) Fine-tuned LFM2.

Figure 6.3.: Comparison of layerwise entropy and layerwise variational entropy profiles (mean) aggregated between known and unknown subsets for LFM2 (baseline) and fine-tuned LFM2 models.

preserve a larger late-layer gap (unknown > known), with the effect most visible for $\bar{V}^{(l)}$.

Answer to RQ1. Yes. Layerwise uncertainty *reliably* distinguishes known from unknown legal questions when summarized over **mid–late layers**, with $\bar{V}^{(l)}$ (green line and purple line) providing the clearest separation. Fine-tuning enlarges the gap between known and unknown uncertainty and makes confidence on in-domain inputs more consistent. In practice, a simple statistic such as $\Delta_{\text{mid-late}}$ (or an averaged $\bar{V}^{(l)}$ over layers 5-8) serves as an effective, architecture-agnostic signal for flagging out-of-domain queries.

6.2.2. Paraphrase Robustness (RQ2)

We assess whether layerwise uncertainty is invariant under meaning-preserving paraphrases of legal questions.

Findings. Across all models and tasks in the known set, paraphrases induce only small, statistically insignificant shifts in layerwise entropy $\bar{H}^{(l)}$ and layerwise variational entropy $\bar{V}^{(l)}$ relative to the *original test-set* instances, particularly in the mid-late layers. Baseline and fine-tuned models follow the same trend, with the fine-tuned models exhibiting slightly greater stability. ECtHR-A and ECtHR-B sometimes show crossings across depth, likely reflecting their closely aligned formulations and tasks; these do not persist and do not affect the overall invariance claim. Detailed per-model and per-task trajectories (means and standard deviations) are provided in Appendix A.

Answer to RQ2. Yes. Layerwise uncertainty is *stable* under meaning-preserving paraphrases, and fine-tuning further enhances this invariance.

6.2.3. Correlations with Accuracy and Macro-F1 (RQ3)

We examine how quantitative characteristics of the uncertainty trajectories (solid lines) connect to task performance. We then relate these statistics to held-out accuracy (CaseHOLD) and macro-F1 (ECtHR-A/B) under the same prompting protocol as in Subsection 6.1.1.

Findings. Two patterns emerge across model families. *(1) Mid–late layers uncertainty tracks errors.* Lower mid–late $\bar{H}^{(l)}$ and especially lower $\bar{V}^{(l)}$ coincide with higher accuracy and macro-F1 on in-domain tasks, while higher mid–late values align with

weaker performance. This mirrors the known-unknown separation from Subsection 6.2.1 and suggests that late consolidation of uncertainty is a reliable indicator of fragile predictions. *(2) Gaps and shape features predictability.* Larger known–unknown gaps $\Delta_{\text{mid-late}}$ and more negative late-layer slopes on known subset associate with higher macro-F1 on ECtHR-A/B and higher accuracy on CaseHOLD. These relationships strengthen with domain specialization (fine-tuning) and are most pronounced for $\bar{V}^{(l)}$.

Answer to RQ3. Quantitative features of the entropy profile, particularly mid-late $\bar{V}^{(l)}$, the late-layer slope, and the known–unknown gap $\Delta_{\text{mid-late}}$, systematically relate to downstream performance. Lower mid–late uncertainty and larger $\Delta_{\text{mid-late}}$ correspond to higher accuracy and macro-F1, with effects magnified under greater domain specificity.

6.2.4. Coupling between Entropy and Variational Entropy (RQ4)

Findings. Across models, entropy $\bar{H}^{(l)}$ and variational entropy $\bar{V}^{(l)}$ are *shape-coupled across depth*: both exhibit co-located mid-late peaks (approximately layers 5-8) and jointly declining tails. This holds for both known (solid lines) and unknown (dash lines) subsets:

- **Solid vs. solid (known).** $\bar{V}^{(l)}$ (green) closely tracks $\bar{H}^{(l)}$ (blue) in shape but remains consistently higher through the mid-late layers; both then decline toward the final layers. Fine-tuning compresses and smooths these trajectories, tightening their alignment and slightly reducing the late-layer VarEntropy-Entropy gap.
- **Dash vs. dash (unknown).** Coupling in shape persists, but $\bar{V}^{(l)}$ *amplifies* the mid-late peak relative to $\bar{H}^{(l)}$ (purple $>$ orange). After fine-tuning, this amplification endures while $\bar{H}^{(l)}$ decreases earlier; $\bar{V}^{(l)}$ stays elevated deeper into the network, preserving a clearer late-layer separation than in baselines.

Early layers show weaker coupling (signals near floor/ceiling), whereas the *strongest coupling* appears in the mid-late layers (coincident peaks, parallel slopes). Fine-tuning consistently *strengthens coupling* on known data (more parallel solid lines) and *increases the sensitivity of* $\bar{V}^{(l)}$ to out-of-domain inputs (larger dashed-line gaps in later layers).

Complementary diagnostics (from Subsection 4.5.6 Boundary Analysis and Control). The pair $(\bar{H}^{(l)}, \bar{V}^{(l)})$ provides interpretable, layerwise regimes:

- *High $\bar{H}^{(l)}$, high $\bar{V}^{(l)}$:* broad uncertainty with a few competing modes; evidence is unstable and the model's commitments can change abruptly.
- *High $\bar{H}^{(l)}$, low $\bar{V}^{(l)}$:* uniformly spread uncertainty; many alternatives remain similarly plausible.
- *Low $\bar{H}^{(l)}$, high $\bar{V}^{(l)}$:* confident but brittle; predictions are concentrated on a small set of divergent outcomes.
- *Low $\bar{H}^{(l)}$, low $\bar{V}^{(l)}$:* confident and stable; the continuation and its local neighborhood agree.

These regimes contextualize the observed coupling: strong mid-late coupling with elevated $\bar{V}^{(l)}$ on unknown inputs indicates confident *but brittle* commitments, whereas fine-tuning shifts known inputs toward the *low-low* regime in later layers.

Answer to RQ4. Yes. $\bar{H}^{(l)}$ and $\bar{V}^{(l)}$ are strongly coupled across models, especially in the mid-late layers that drive known/unknown discrimination. $\bar{V}^{(l)}$ behaves as a *sharpened companion*: it follows the depthwise shape of $\bar{H}^{(l)}$ but magnifies uncertainty on unknown inputs and remains elevated later in depth. Fine-tuning tightens the coupling for known inputs while preserving a larger VarEntropy-Entropy gap for unknown inputs in the final layers.

6.2.5. Architecture Sensitivity (RQ5)

We assess the extent to which layerwise uncertainty depends on model family. Using the setup of Subsection 6.1.2, we compare the depthwise profiles of entropy ($\bar{H}^{(l)}$) and variational entropy ($\bar{V}^{(l)}$) across the three model families evaluated (baseline vs. fine-tuned variants).

Findings. Across families, we observe *shared qualitative structure* with *architecture-specific geometry*:

- **Peak position and scale vary by family.** The *layer index* of the peak and its *magnitude* shift across models: Llama family shows broader, higher-magnitude peaks that drop sharply toward the output layers; Zamba exhibits lower absolute $\bar{H}^{(l)}$ but a pronounced $\bar{V}^{(l)}$ spike in mid depth; LFM2 displays a steeper rise and later decline, with sharper mid-late separation.

- **Effect of fine-tuning is consistent but family-modulated.** Fine-tuning universally *suppresses* known-subset uncertainty earlier in depth and *preserves/elevates* unknown-subset uncertainty deeper into the network, especially for $\bar{V}^{(l)}$. The extent of this sharpening depends on the family.

Answer to RQ5. *Moderately architecture-sensitive.* The diagnostic value of layerwise uncertainty is *qualitatively robust* across model families in the mid–late layers consistently separate known from unknown and $\bar{V}^{(l)}$ is the most discriminative. Fine-tuning enhances these effects uniformly, tightening known subset profiles and enlarging unknown-subset gaps, with the strongest late-layer separation in families that develop sharper mid-late peaks.

7. Discussion

7.1. Near-OOD and Cross-Domain Task Similarities

A central question in this study is how models behave on inputs that are *near-OOD*: examples that differ from the fine-tuning distribution yet remain within closely related legal subdomains (Chalkidis, Jana, Hartung, et al. 2022). In this respect, UNFAIR-ToS occupies an interesting position relative to CaseHOLD and ECtHR-A/B. Although UNFAIR-ToS is reserved for unknown-set evaluation in our protocol, it shares salient structural properties with the primary tasks: a standardized, lettered multi-choice interface, and legal semantics that remain adjacent to the doctrinal reasoning elicited by ECtHR-A/B and the doctrinal matching demanded by CaseHOLD. These commonalities suggest that, for models that neither overfit nor underfit, UNFAIR-ToS functions as a near-OOD probe rather than a fully unrelated domain shift. The empirical layerwise profiles support this reading.

7.2. ECtHR-A vs. ECtHR-B: A Source of Label-Semantics Confusion

ECtHR-A labels the *violations found* by the Court, whereas ECtHR-B labels the *alleged provisions* put before the Court irrespective of the outcome. While both tasks present the same factual paragraphs and share the same A–J article legend, their label semantics diverge. This creates a potential locus of confusion for models: identical facts, identical optionization, but different target mappings.

Our layerwise perspective clarifies how such confusion may manifest. Because the surface form and option legend remain fixed, early-layer uncertainty can be deceptively low, while mid-late layers show higher varentropy when the model is forced to disambiguate between allegation and outcome.

7.3. LEDGAR as a Hard Task: Entropy Geometry Across Models

LEDGAR presents a challenging single-label classification problem with 100 provision types. The very large label space, combined with contractual paragraphs, increases the

intrinsic ambiguity of next-token predictions under unified multiple-choice question interface. From an information-theoretic standpoint, this inflate the space of mutually exclusive alternatives and thereby tends to elevate the entropy baseline, especially for models not explicitly adapted to contract-style drafting.

Consistent with this difficulty, the aggregated unknown-set profiles, to which LEDGAR contributes, show that mid-late layers maintain higher entropy and markedly higher variational entropy than known tasks for both baseline and fine-tuned models. Notably, while fine-tuning compresses and smooths known-task trajectories, it does not fully eliminate the elevated variational entropy on unknown inputs in later layers. This produces a characteristic high-variational entropy tail that appears across architectures, indicating that LEDGAR remains hard even when the model's in-domain confidence improves.

7.4. Selective State Space Effects in Mamba S6

Figures 6.2 and 6.3 compare layerwise entropy and layerwise variational entropy for Zamba2 (Glorioso, Anthony, Tokpanov, et al. 2024; Gu and Dao 2024) and LFM2 (Thomas, Parnichkun, Amini, et al. 2024) in baseline and fine-tuned settings. In both model families, fine-tuning compresses uncertainty on known inputs while preserving a higher trajectory for unknown inputs in mid-late layers. The effect is especially visible in the variational entropy profiles, which show a more persistent tail for unknown inputs even as the known curves flatten after fine-tuning.

This observation suggests that these patterns are consistent with a selective state space mechanism in Mamba S6 that can strengthen separation between known and unknown inputs in deeper layers. Selective scanning and gating can encourage stable commitments on in-domain examples while maintaining sensitivity to distributional shifts.

By contrast, Transformer-only backbones such as LLaMA (Grattafiori, Dubey, Jauhri, et al. 2024; Vaswani, Shazeer, Parmar, et al. 2023) rely solely on attention mechanism without the same selective state update. While attention mechanism can also yield late-layer separation, our results indicate that Mamba S6 may widen the known-unknown gap more clearly in mid-late layers.

7.5. Attention Mask is Needed During Inference for Zamba2

Although causal mask prevents looking ahead in a sequence. A padding mask identifies which token positions are real and which are padding. Selective SSMs such as Zamba2

are inherently causal by construction, but they still *must* know which positions are padding so that the internal recurrence does not keep evolving on padded tails.

Discrete SSM update and why padding matters. Let the continuous state space system be

$$\dot{h}(t) = Ah(t) + Bu(t), \qquad y(t) = Ch(t),$$

with hidden state h, input u, and output y. Under ZOH with per step duration Δt_k, the discrete index $k \in \{1, \dots, T\}$ denotes the step in the left to right scan (equivalently, the token position in the sequence). The discrete update is

$$\bar{A}(\Delta t_k) = e^{\Delta t_k A}, \qquad \bar{B}(\Delta t_k) = \int_0^{\Delta t_k} e^{\tau A}\, d\tau\, B = A^{-1}\big(e^{\Delta t_k A} - I\big)\, B,$$

$$h_k = \bar{A}(\Delta t_k)\, h_{k-1} + \bar{B}(\Delta t_k)\, u_k, \qquad y_k = C\, h_k.$$

When $\Delta t_k = 0$, we have $\bar{A}(0) = I$ and $\bar{B}(0) = 0$. Intuitively, if no time elapses there is no state change.

Key takeaway. Passing an attention mask tells a selective SSM when to *carry* its state unchanged on padding. Without this signal, the model keeps evolving through padding tokens, which can degrade quality and can trigger numerical instability in deeper layers.

7.6. PEFT for Mamba S6 is Less Mature than for Transformer Models

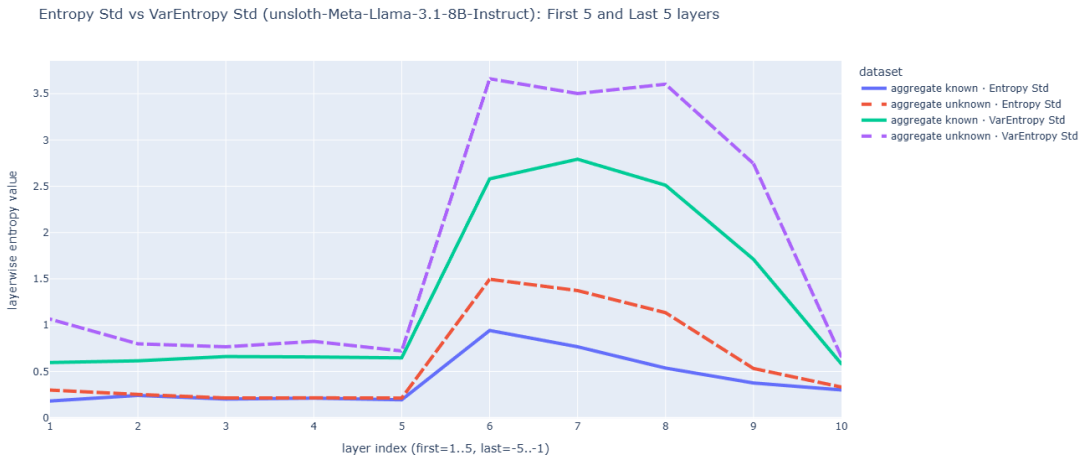
PEFT for Transformer backbones is well established, with LoRA, adapters, and related techniques commonly targeting attention and feed forward projections. In contrast, PEFT for Mamba S6 style selective SSMs is not yet as fully developed. The main reason is architectural: Mamba style layers combine input and output projections with a selective state update that is implemented through discretized state space parameters and scan operations. Thus, many off-shelf PEFT methods are designed for attention blocks and do not directly address the internal SSM parameterization.

Emerging approaches. Recent works on MambaPEFT (Yoshimura, Hayashi, and Maeda 2025) and SSM-PEFT (Galim, Kang, Zeng, et al. 2025) explore PEFT for Mamba-style models. Approaches include placing low-rank adapters on SSM specific components, introducing small bottleneck modules to modulate the selective update, and sharing low rank factors across layers to control memory cost. This area is evolving, but it is not yet as standardized as PEFT for Transformers.

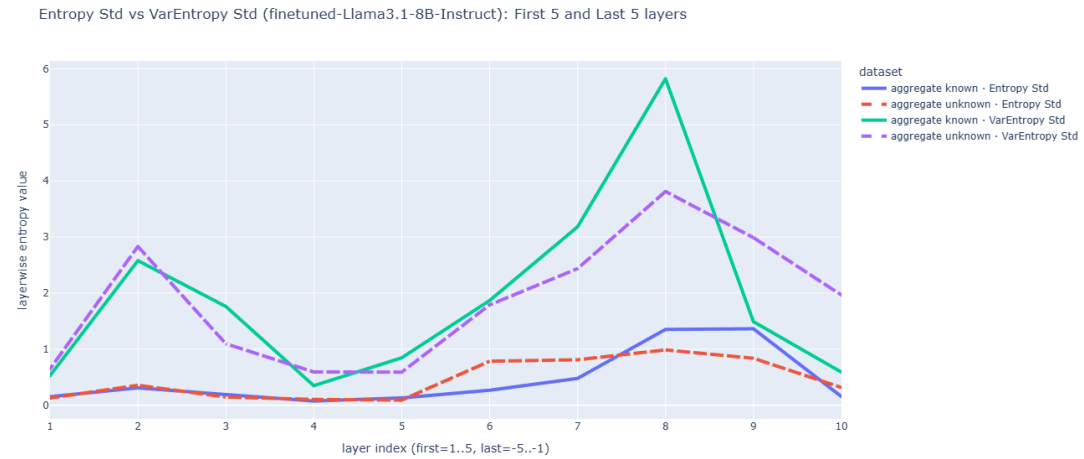
8. Conclusion

In conclusion, this thesis introduces layerwise entropy as a structural lens for assessing LLM reliability and shows that it separates known from unknown inputs. The analysis links uncertainty dynamics within the network to truthfulness (S. Lin, Hilton, and Evans 2022) and faithfulness (Ji, Lee, Frieske, et al. 2023) criteria and demonstrates consistent gains after domain specialization, providing an internal signal that is both interpretable and practical for selective prediction, deferral, and verification in knowledge-intensive settings. We therefore identify layerwise entropy as a direction for future work.

A. Additional Uncertainty Figures

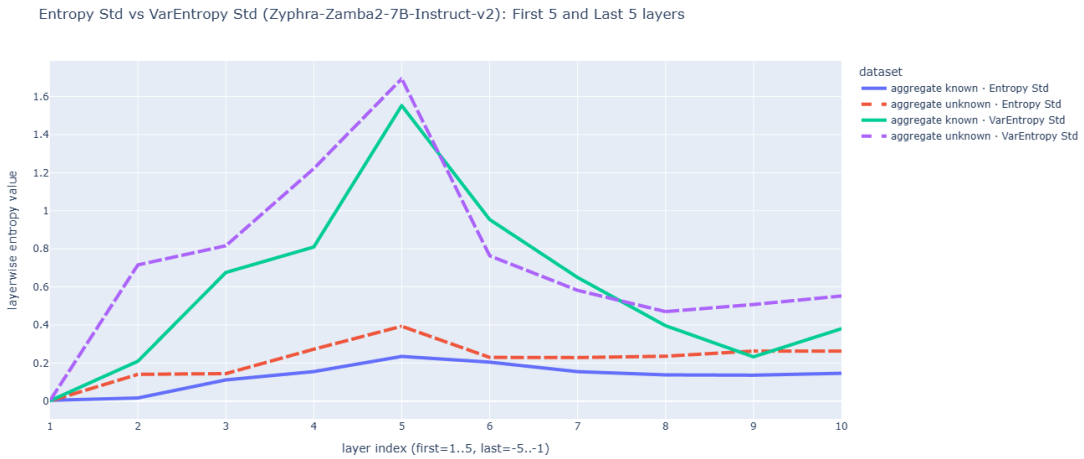


(a) Llama-3.1-8B-Instruct (baseline).

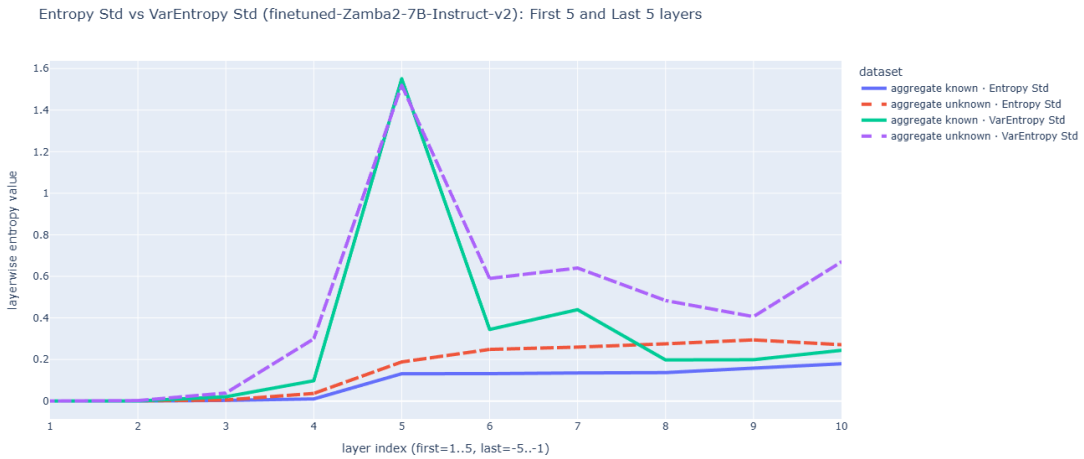


(b) Fine-tuned Llama-3.1.

Figure A.1.: Comparison of layerwise entropy and layerwise variational entropy profiles (standard deviation) aggregated between known and unknown subsets for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models.

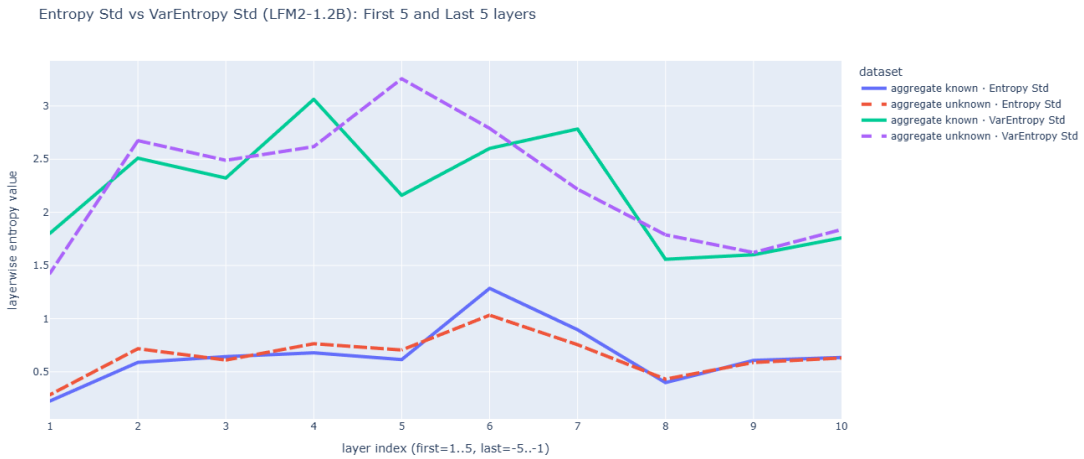


(a) Zamba2-7B-Instruct-v2 (baseline).

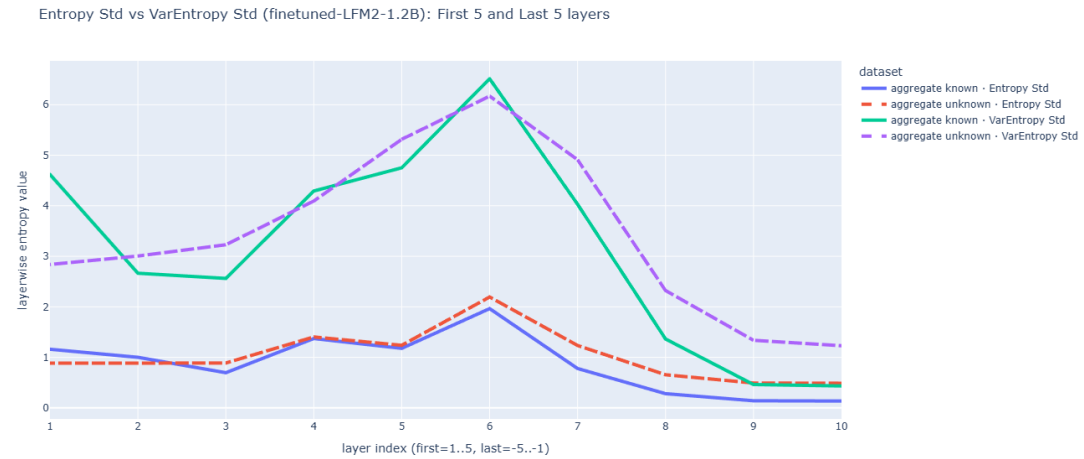


(b) Fine-tuned Zamba2.

Figure A.2.: Comparison of layerwise entropy and layerwise variational entropy profiles (standard deviation) aggregated between known and unknown subsets for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models.

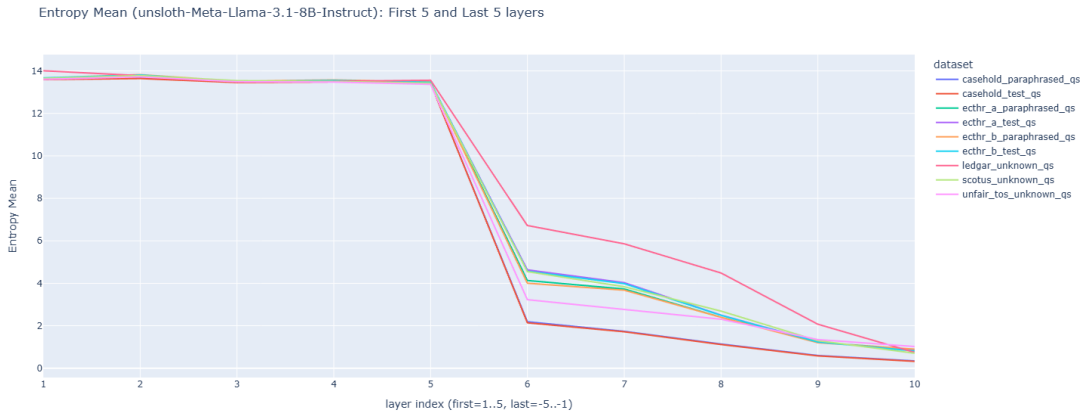


(a) LFM2 (baseline).

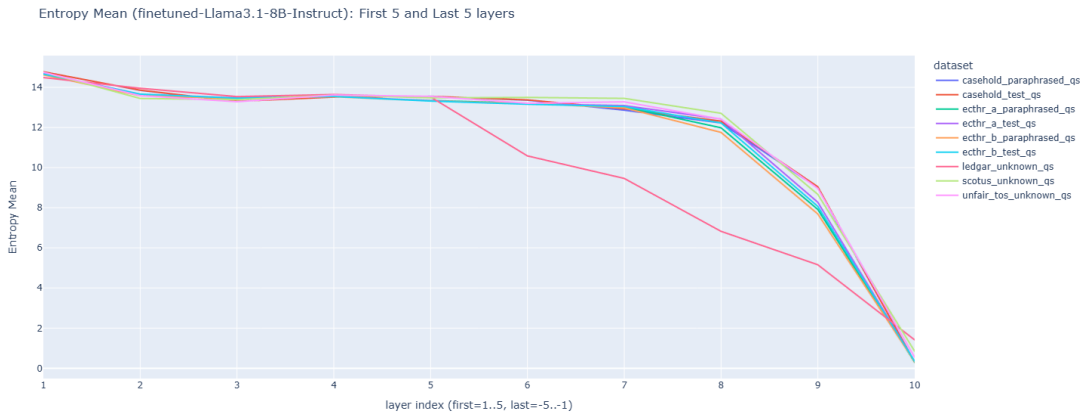


(b) Fine-tuned LFM2.

Figure A.3.: Comparison of layerwise entropy and layerwise variational entropy profiles (standard deviation) aggregated between known and unknown subsets for LFM2 (baseline) and fine-tuned LFM2 models.

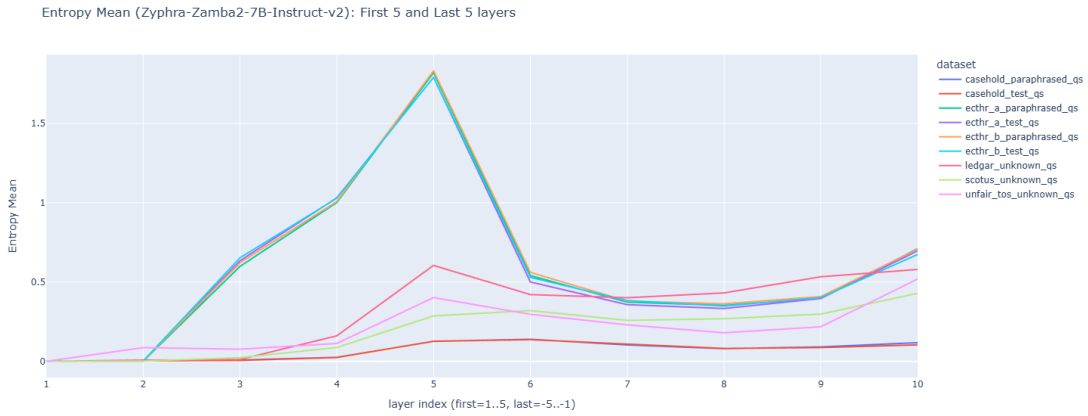


(a) Llama-3.1-8B-Instruct (baseline).

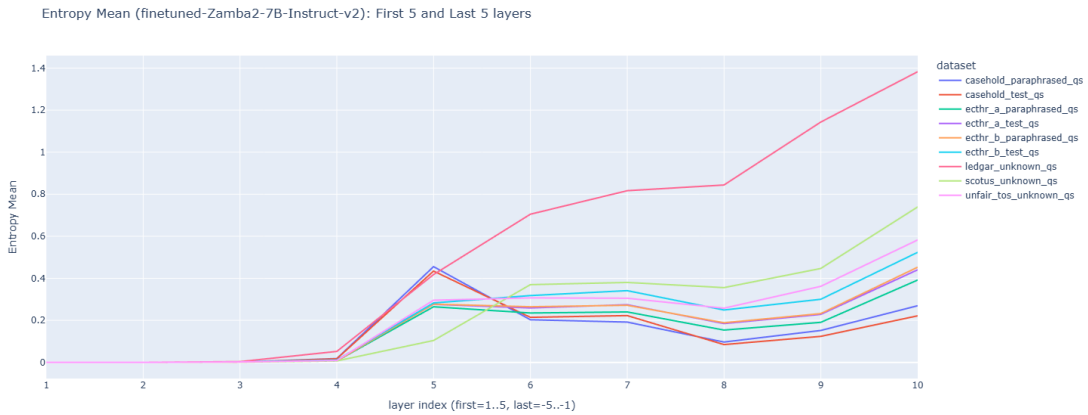


(b) Fine-tuned Llama-3.1.

Figure A.4.: Mean of layerwise entropy across depth for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models. Each line corresponds to a single task/dataset split.

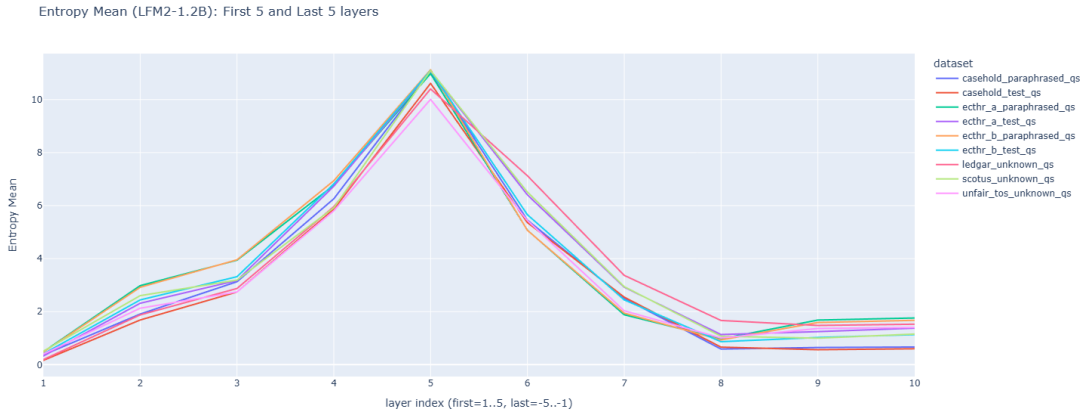


(a) Zamba2-7B-Instruct-v2 (baseline).

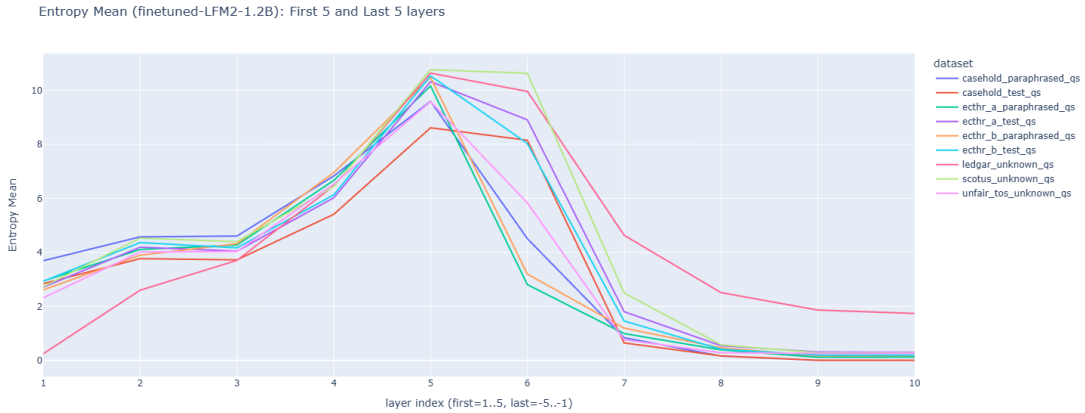


(b) Fine-tuned Zamba2.

Figure A.5.: Mean of layerwise entropy across depth for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models. Each line corresponds to a single task/dataset split.

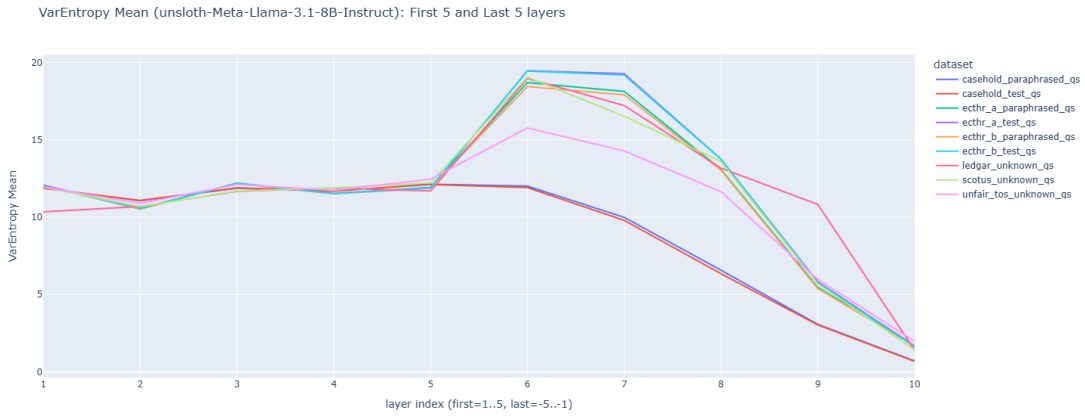


(a) LFM2 (baseline).

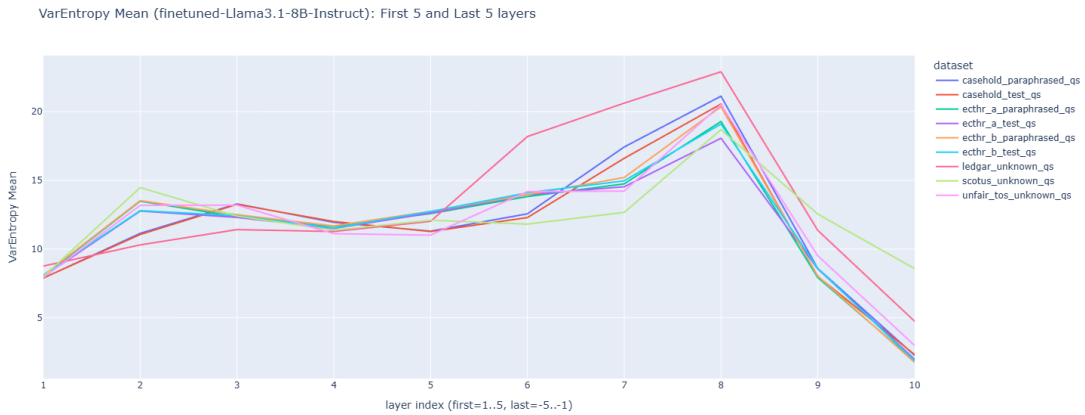


(b) Fine-tuned LFM2.

Figure A.6.: Mean of layerwise entropy across depth for LFM2 (baseline) and fine-tuned LFM2 models. Each line corresponds to a single task/dataset split.

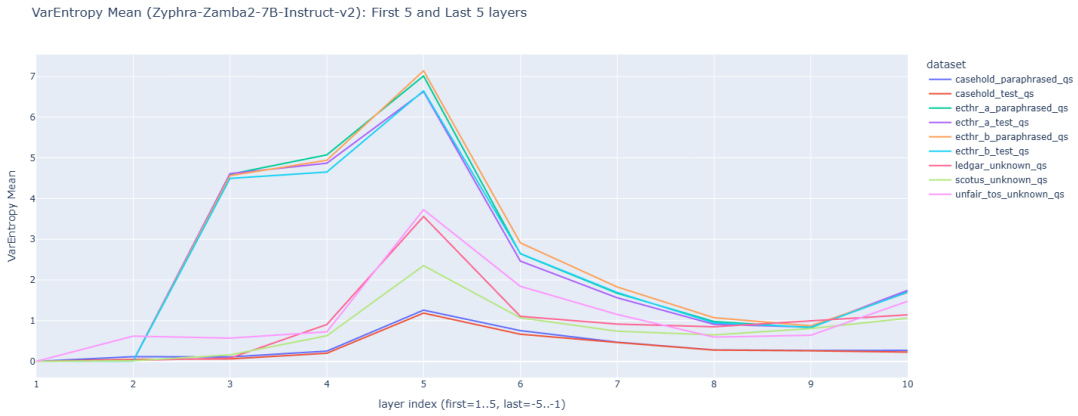


(a) Llama-3.1-8B-Instruct (baseline).

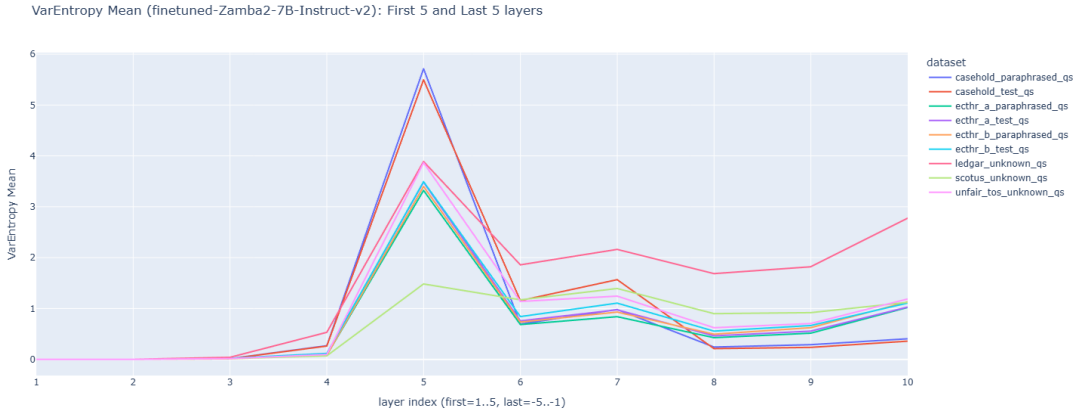


(b) Fine-tuned Llama-3.1.

Figure A.7.: Mean of layerwise variational entropy across depth for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models. Each line corresponds to a single task/dataset split.

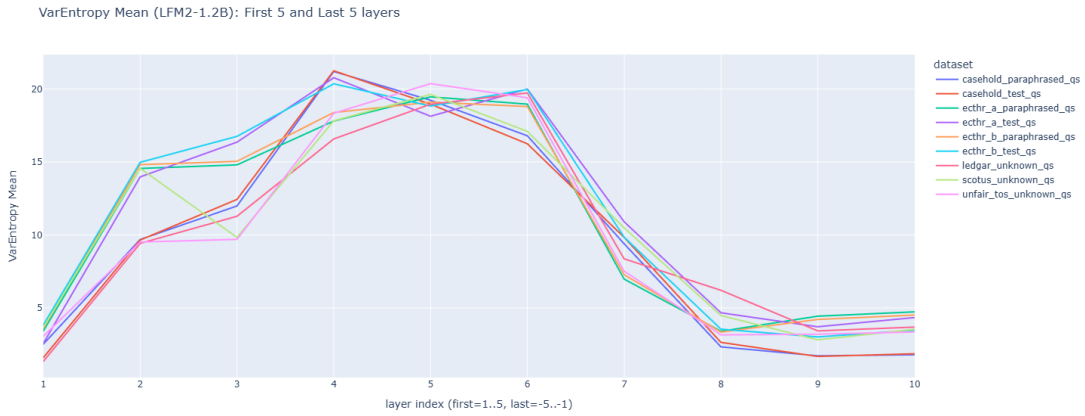


(a) Zamba2-7B-Instruct-v2 (baseline).

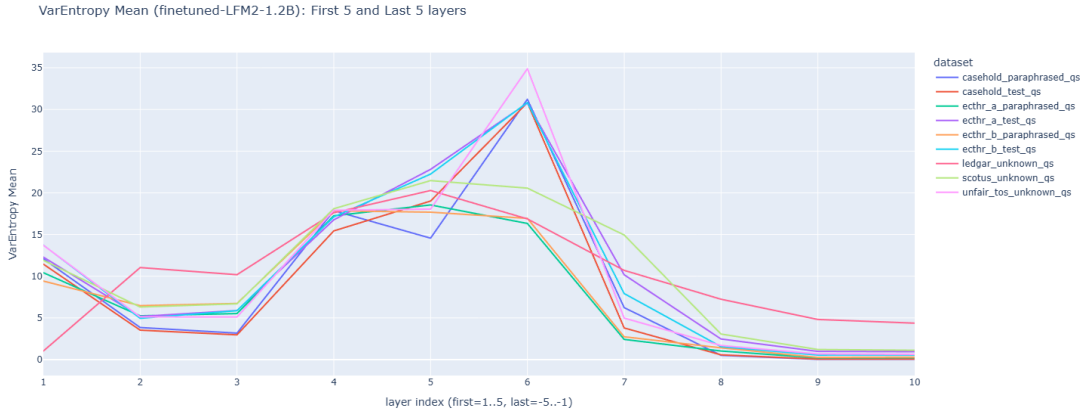


(b) Fine-tuned Zamba2.

Figure A.8.: Mean of layerwise variational entropy across depth for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models. Each line corresponds to a single task/dataset split.

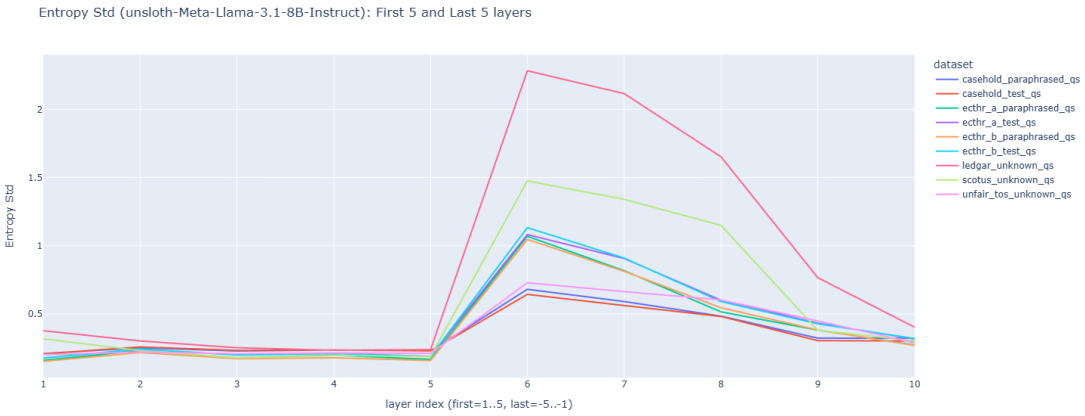


(a) LFM2 (baseline).

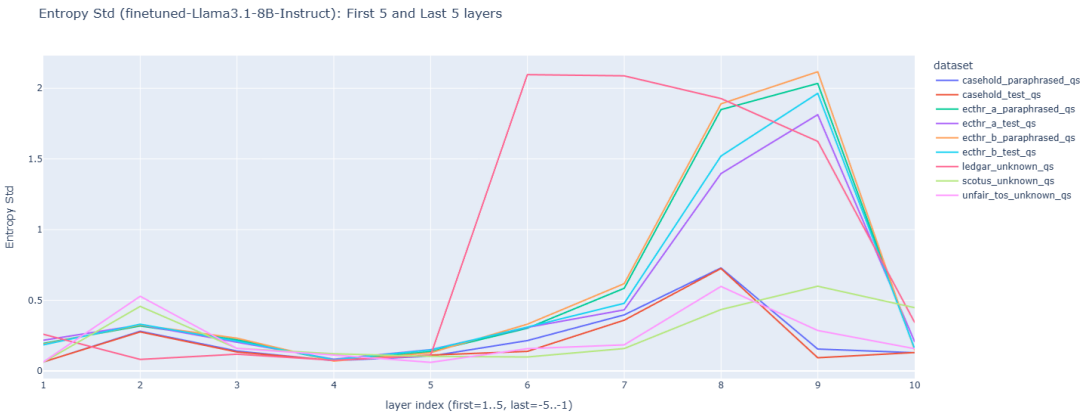


(b) Fine-tuned LFM2.

Figure A.9.: Mean of layerwise variational entropy across depth for LFM2 (baseline) and fine-tuned LFM2 models. Each line corresponds to a single task/dataset split.

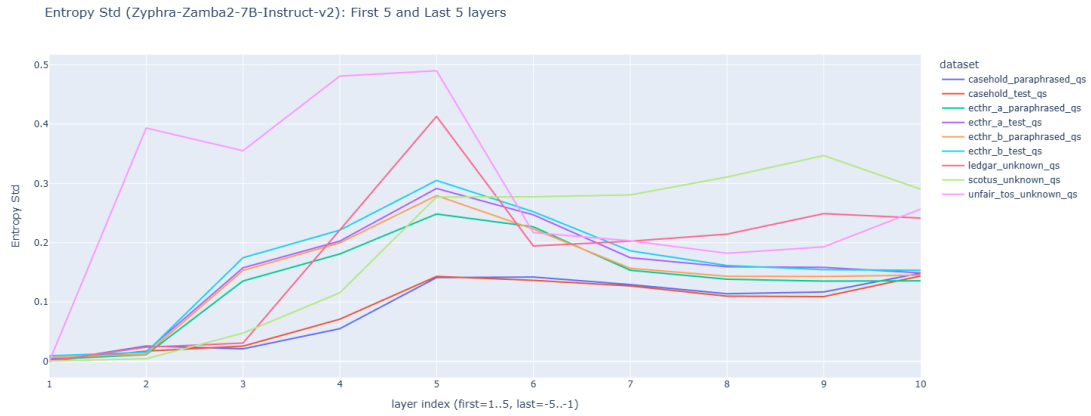


(a) Llama-3.1-8B-Instruct (baseline).

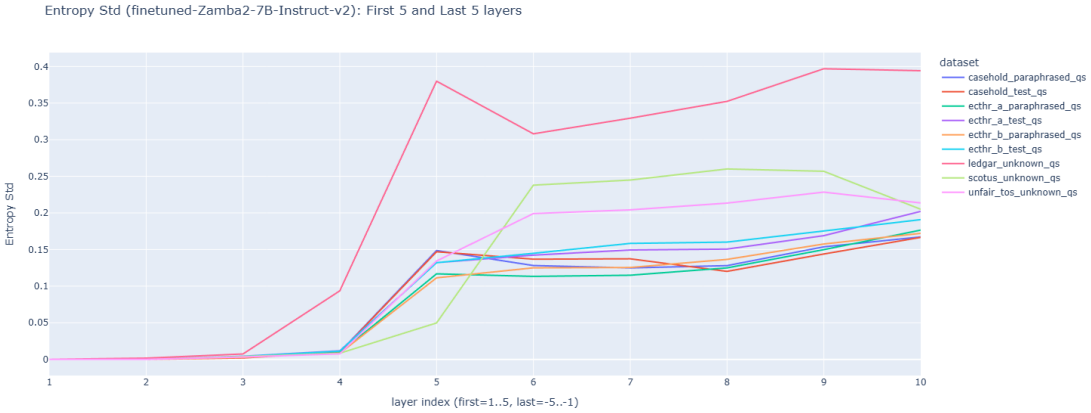


(b) Fine-tuned Llama-3.1.

Figure A.10.: Standard deviation of layerwise entropy across depth for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models. Each line corresponds to a single task/dataset split.

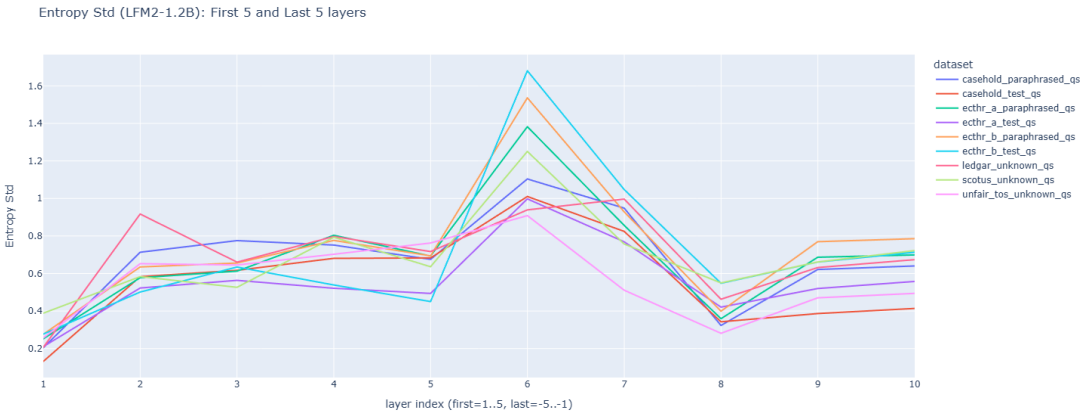


(a) Zamba2-7B-Instruct-v2 (baseline).



(b) Fine-tuned Zamba2.

Figure A.11.: Standard deviation of layerwise entropy across depth for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models. Each line corresponds to a single task/dataset split.

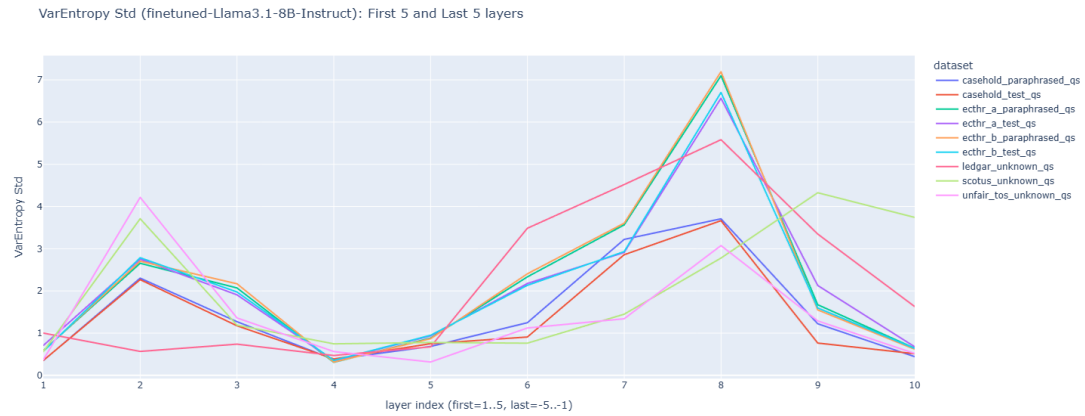


(a) LFM2 (baseline).

(b) Fine-tuned LFM2.

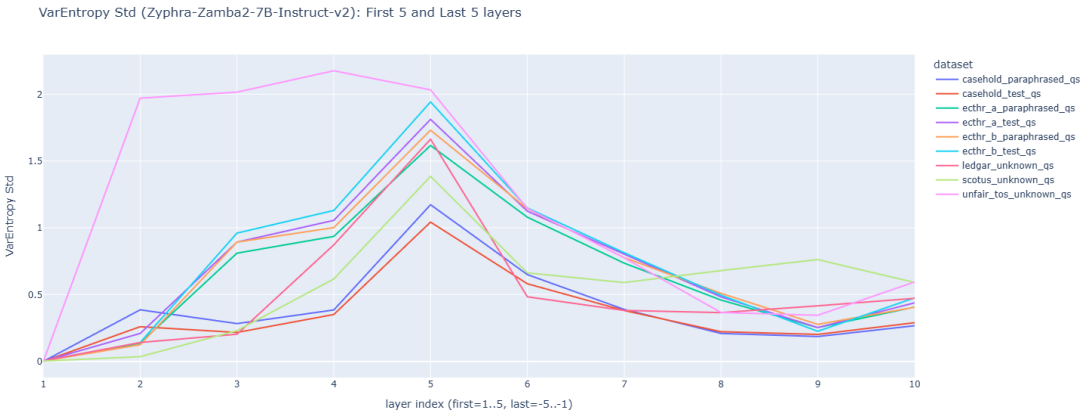
Figure A.12.: Standard deviation of layerwise entropy across depth for LFM2 (baseline) and fine-tuned LFM2 models. Each line corresponds to a single task/-dataset split.

(a) Llama-3.1-8B-Instruct (baseline).

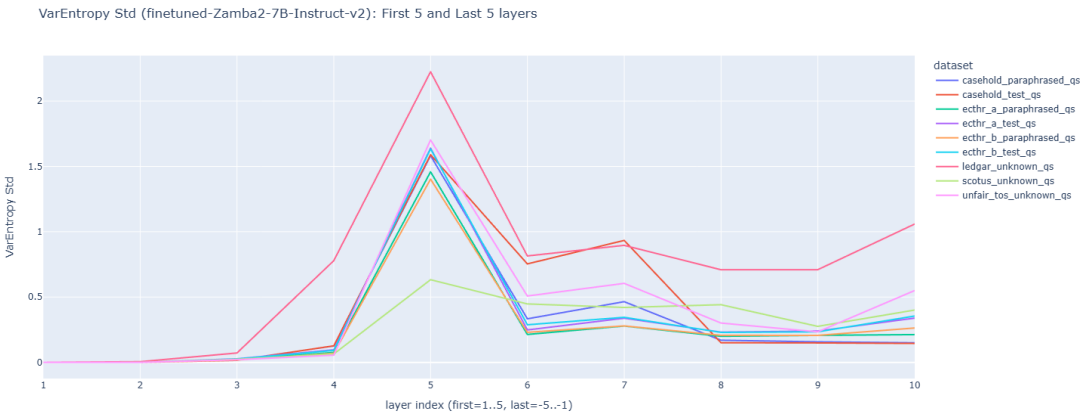


(b) Fine-tuned Llama-3.1.

Figure A.13.: Standard deviation of layerwise variational entropy across depth for Llama-3.1-8B-Instruct (baseline) and fine-tuned Llama-3.1 models. Each line corresponds to a single task/dataset split.



(a) Zamba2-7B-Instruct-v2 (baseline).



(b) Fine-tuned Zamba2.

Figure A.14.: Standard deviation of layerwise variational entropy across depth for Zamba2-7B-Instruct-v2 (baseline) and fine-tuned Zamba2 models. Each line corresponds to a single task/dataset split.

(a) LFM2 (baseline).

(b) Fine-tuned LFM2.

Figure A.15.: Standard deviation of layerwise variational entropy across depth for LFM2 (baseline) and fine-tuned LFM2 models. Each line corresponds to a single task/dataset split.

Abbreviations

NLP Natural Language Processing

SOTA state-of-the-art

LLM Large Language Model

STAR Synthesis of Tailored Architectures

LIV Linear Input-Varying

QA Question Answering

PEFT Parameter-Efficient Fine-Tuning

LoRA Low-Rank Adaptation

MLP Multi-Layer Perceptron

FLOP Floating-Point Operation

OOD out-of-distribution

SSM State Space Model

ZOH Zero-Order Hold

FFT Fast Fourier Transform

LTI Linear Time–Invariant

RNN Recurrent Neural Network

DC Direct Current

SVD Singular Value Decomposition

GLUE General Language Understanding Evaluation

MMLU Massive Multitask Language Understanding

BLEU Bilingual Evaluation Understudy

ROUGE Recall-Oriented Understudy for Gisting Evaluation

METEOR Metric for Evaluation of Translation with Explicit ORdering

LexGLUE Legal General Language Understanding Evaluation

NLU Natural Language Understanding

ECtHR European Court of Human Rights

ECHR European Convention of Human Right

LEDGAR Labeled Electronic Data Gathering, Analysis, and Retrieval system

SEC Securities and Exchange Commission

ToS Terms of Service

SCOTUS U.S. Supreme Court

AMP Automatic Mixed Precision

ICL In-Context Learning

RoPE Rotary Positional Embedding

RAG Retrieval-Augmented Generation

LFM Liquid Foundation Model

SFT Supervised Fine-Tuning

BERT Bidirectional Encoder Representations from Transformers

DIPPER Discourse Paraphraser

List of Figures

Bibliography

[1] S. Abnar, L. Beinborn, R. Choenni, and W. Zuidema. "Blackbox Meets Blackbox: Representational Similarity & Stability Analysis of Neural Language Models and Brains." In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by T. Linzen, G. Chrupała, Y. Belinkov, and D. Hupkes. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 191–203. DOI: 10.18653/v1/W19-4820.

[2] S. Banerjee and A. Lavie. "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments." In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ed. by J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 65–72.

[3] I. Chalkidis, A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. M. Katz, and N. Aletras. *LexGLUE: A Benchmark Dataset for Legal Language Understanding in English*. 2022. arXiv: 2110.00976 [cs.CL].

[4] S. Chen, M. Xiong, J. Liu, Z. Wu, T. Xiao, S. Gao, and J. He. *In-Context Sharpness as Alerts: An Inner Representation Perspective for Hallucination Mitigation*. 2024. arXiv: 2403.01548 [cs.CL].

[5] S. F. Chen and J. Goodman. "An empirical study of smoothing techniques for language modeling." In: *Computer Speech & Language* 13.4 (1999), pp. 359–394. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1999.0128.

[6] Q. Cheng, T. Sun, X. Liu, W. Zhang, Z. Yin, S. Li, L. Li, Z. He, K. Chen, and X. Qiu. *Can AI Assistants Know What They Don't Know?* 2024. arXiv: 2401.13275 [cs.CL].

[7] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. *What Does BERT Look At? An Analysis of BERT's Attention*. 2019. arXiv: 1906.04341 [cs.CL].

[8] Entropix. *Entropy Based Sampling and Parallel CoT Decoding*. 2024. URL: https://github.com/xjdr-alt/entropix.

[9] K. Galim, W. Kang, Y. Zeng, H. I. Koo, and K. Lee. "Parameter-Efficient Fine-Tuning of State Space Models." In: *Forty-second International Conference on Machine Learning*. 2025.

[10] P. Glorioso, Q. Anthony, Y. Tokpanov, A. Golubeva, V. Shyam, J. Whittington, J. Pilault, and B. Millidge. *The Zamba2 Suite: Technical Report*. 2024. arXiv: 2411.15242 [cs.LG].

[11] A. Grattafiori, A. Dubey, A. Jauhri, et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI].

[12] A. Gu and T. Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. 2024. arXiv: 2312.00752 [cs.LG].

[13] A. Gu, K. Goel, and C. Ré. *Efficiently Modeling Long Sequences with Structured State Spaces*. 2022. arXiv: 2111.00396 [cs.LG].

[14] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[15] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. *Measuring Massive Multitask Language Understanding*. 2021. arXiv: 2009.03300 [cs.CY].

[16] J. Hewitt and C. D. Manning. "A Structural Probe for Finding Syntax in Word Representations." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4129–4138. DOI: 10.18653/v1/N19-1419.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[18] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions." In: *ACM Transactions on Information Systems* 43.2 (Jan. 2025), pp. 1–55. ISSN: 1558-2868. DOI: 10.1145/3703155.

[19] S. Jain and B. C. Wallace. *Attention is not Explanation*. 2019. arXiv: 1902.10186 [cs.CL].

[20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. "Survey of Hallucination in Natural Language Generation." In: *ACM Computing Surveys* 55.12 (Mar. 2023), pp. 1–38. ISSN: 1557-7341. DOI: 10.1145/3571730.

[21] K. Krishna, Y. Song, M. Karpinska, J. Wieting, and M. Iyyer. *Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense*. 2023. arXiv: 2303.13408 [cs.CL].

[22] L. Kuhn, Y. Gal, and S. Farquhar. *Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation*. 2023. arXiv: 2302.09664 [cs.CL].

[23] B. Lakshminarayanan, A. Pritzel, and C. Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. 2017. arXiv: 1612.01474 [stat.ML].

[24] C.-Y. Lin. "ROUGE: A Package for Automatic Evaluation of Summaries." In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81.

[25] S. Lin, J. Hilton, and O. Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: 2109.07958 [cs.CL].

[26] A. Malinin and M. Gales. *Uncertainty Estimation in Autoregressive Structured Prediction*. 2021. arXiv: 2002.07650 [stat.ML].

[27] N. Mathur, T. Baldwin, and T. Cohn. "Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 4984–4997. DOI: 10.18653/v1/2020.acl-main.448.

[28] M. Mountantonakis and Y. Tzitzikas. *Validating ChatGPT Facts through RDF Knowledge Graphs and Sentence Similarity*. 2023. arXiv: 2311.04524 [cs.DB].

[29] OptimLLM. *Entropy decoding implementations for LLMs*. 2024. URL: https://github.com/algorithmicsuperintelligence/optillm.

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. "Bleu: a Method for Automatic Evaluation of Machine Translation." In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Ed. by P. Isabelle, E. Charniak, and D. Lin. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[31] J. Park, J. Park, Z. Xiong, N. Lee, J. Cho, S. Oymak, K. Lee, and D. Papailiopoulos. *Can Mamba Learn How to Learn? A Comparative Study on In-Context Learning Tasks*. 2024. arXiv: 2402.04248 [cs.LG].

[32] N. Sadvilkar and M. Neumann. "PySBD: Pragmatic Sentence Boundary Disambiguation." In: *Proceedings of the 2nd Workshop for NLP Open Source Software (NLP-OSS)*. 2020.

[33] C. E. Shannon. "A mathematical theory of communication." In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[34] G. Sriramanan, S. Bharti, V. S. Sadasivan, S. Saha, P. Kattakinda, and S. Feizi. "LLM-Check: Investigating Detection of Hallucinations in Large Language Models." In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024.

[35] I. Tenney, D. Das, and E. Pavlick. *BERT Rediscovers the Classical NLP Pipeline*. 2019. arXiv: 1905.05950 [cs.CL].

[36] A. W. Thomas, R. Parnichkun, A. Amini, S. Massaroli, and M. Poli. *STAR: Synthesis of Tailored Architectures*. 2024. arXiv: 2411.17800 [cs.LG].

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[38] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 2020. arXiv: 1905.00537 [cs.CL].

[39] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[40] M. Yoshimura, T. Hayashi, and Y. Maeda. "MambaPEFT: Exploring Parameter-Efficient Fine-Tuning for Mamba." In: *The Thirteenth International Conference on Learning Representations*. 2025.

[41] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL].