Does Representation Matter? Exploring Intermediate Layers in Large Language Models

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Understanding what constitutes a "good" representation in large language models (LLMs) is a fundamental question in natural language processing. In this paper, we investigate the quality of representations at different layers of LLMs, specifically Transformers and State Space Models (SSMs). We find that intermediate layers consistently provide better representations for downstream tasks compared to final layers. To quantify representation quality, we employ existing metrics from other contexts—such as prompt entropy, curvature, and augmentation-invariance—and apply them to LLMs. Our experiments reveal significant differences between architectures, showcase how representations evolve during training, and illustrate the impact of input randomness and prompt length on different layers. Notably, we observe a bimodal behavior in entropy within intermediate layers and explore potential causes related to training data exposure. Our findings offer valuable insights into the internal workings of LLMs and open avenues for optimizing their architectures and training processes.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing by achieving remarkable performance across a variety of tasks (Muennighoff et al., 2022; Hendrycks et al., 2020). Despite their success, understanding what constitutes a "good" representation within these models remains an open question. Specifically, how do representations at different layers contribute to downstream task performance, and how can we quantify their quality?

Most previous studies have primarily focused on final-layer representations, often overlooking the potential of intermediate layers. However, recent work suggests that intermediate layers may offer richer or more generalizable features for certain tasks (Bordes et al., 2022; Fan et al., 2024). This observation prompts a deeper investigation into the layerwise behavior of LLMs.

In this paper, we explore the quality of representations across different layers of LLMs in various settings, including different model architectures (Transformers (Vaswani, 2017) vs. State Space Models (SSMs) (Gu & Dao, 2023)), training checkpoints, input randomness, and prompt length. Our contributions are threefold:

- We demonstrate that intermediate layers consistently yield much better representations for downstream tasks than final layers.
- We apply and adapt existing metrics—such as prompt entropy, curvature, and augmentation-invariance—to quantify representation quality in LLMs.
- We analyze how these metrics vary across different settings, including architectural differences (Transformers vs. SSMs), training progression, input randomness, and prompt length.

Table 1: MTEB Downstream Task Performance Using Representations from Different Layers

Model	Number of Tasks where Best Performance is not in Last Layer	Avg. Last Layer Performance	Avg. Best Layer Performance
LLM2Vec 8B (Transformer)	100%	64.7%	66.8%
Pythia 410M (Transformer)	96.6%	49.8%	53.3%
Mamba 130M (SSM)	100%	46.9%	50.9%

Furthermore, we uncover significant differences in the behavior of these metrics between Transformers and SSMs. For example, we observe a bimodal distribution in entropy within intermediate layers and investigate potential causes, such as the influence of training data examples.

Our findings provide new insights into the internal mechanisms of LLMs and offer practical guidance for model selection and architectural design in future research.

2 Related Work

Understanding representations in neural networks has been a topic of extensive research. Alain (2016) analyzed hidden representations to interpret neural networks' learning processes. Raghu et al. (2017) introduced Singular Vector Canonical Correlation Analysis to compare representations across layers and networks. In the context of Transformers, Liu et al. (2019) studied the linguistic knowledge captured at different layers, finding that lower layers encode more syntactic information while higher layers capture semantic features. A similar work (Jin et al., 2024) showed that semantic concepts are learned in intermediate layers. They proposed a layerwise probing technique to discover exactly in which layer concepts are formed. On the other hand, state-space models have been less explored in this regard. Gu & Dao (2023) introduced MAMBA, an SSM architecture capable of handling long sequences efficiently. However, comparative studies between SSMs and Transformers at the representation level remain scarce.

Metrics like entropy and curvature have been used in other contexts to analyze representations. Shwartz-Ziv & Tishby (2017); Shwartz-Ziv (2022) discussed the Information Bottleneck principle, suggesting that networks learn to compress representations. Hosseini & Fedorenko (2024) introduced curvature as a measure of representational dynamics in recurrent networks. Several works in the vision domain proposed unsupervised representational quality metrics that are strongly correlated with accuracy on downstream tasks (Garrido et al., 2023; Agrawal et al., 2022; Thilak et al., 2023). The RankMe measure can be shown to be a measure of entropy known as matrix-based entropy, which we use in our subsequent analysis.

Our work bridges these areas by applying and adapting such metrics to LLMs, providing a novel perspective on representation quality across architectures and training stages.

3 Methodology

3.1 Definitions

Let $\mathbf{Z} \in \mathbb{R}^{N \times D}$ represent a batch of N samples, each with dimensionality D. The vector z_i denotes the i-th row of Z. We denote the i-th largest eigenvalue of a matrix $\mathbf{M}$ as $\lambda_i(\mathbf{M})$, and the trace of $\mathbf{M}$ by $\operatorname{tr}(\mathbf{M})$. Input sequences are denoted by $\mathbf{x} \in \mathbb{R}^{L \times d}$ and output sequences by $\mathbf{y} \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the feature dimension.

3.2 Architectures

In this study, we compare two prominent architectures: Transformer-based models (Vaswani, 2017) and State Space Models (SSMs) (Gu & Dao, 2023). Transformers utilize self-attention mechanisms to capture long-range dependencies within input sequences, enabling parallel processing and effective encoding of complex patterns. On the other hand, SSMs employ recurrent dynamics to handle sequential information with linear time and memory complexity, offering efficiency in processing longer sequences. Despite their differing approaches, both architectures aim to generate rich and meaningful representations across multiple layers. For detailed mathematical formulations and parameter configurations of each architecture, please refer to Appendix C.

3.3 Representation Evaluation Metrics

To quantify the quality of representations across layers, we employ two categories of metrics: token embedding diversity metrics and augmentation-invariance metrics. We rigorously introduce each of the following metrics in Appendix B.

Token embedding diversity metrics evaluate the variability and richness of the representations at the token level within a single sequence. We employ prompt entropy (Wei et al., 2024) and curvature (Hosseini & Fedorenko, 2024). Of particular interest is the prompt entropy, which measures the amount of compression in a prompt's token representations.

Augmentation-invariance metrics assess the robustness of representations to augmentations on the input prompt. We employ DiME (Skean et al., 2023), infoNCE (Oord et al., 2018), and LiDAR (Thilak et al., 2023). We provide full details and examples of the augmentation process in Appendix F.

4 Experiments

4.1 Intermediate Layers Yield Better Representations for Downstream Embedding Tasks

First, we evaluate the performance of the representations of each layer in downstream tasks in the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2022). This benchmark is designed to test the performance of LLMs on various embedded tasks. We chose 32 tasks that range from classification, clustering, and re-ranking. We evaluated each layer of Pythia 410M, Mamba 130M, and LLM2Vec-unsup-simcse (BehnamGhader et al., 2024).

Interestingly, the intermediate layers consistently outperform the final layers in all architectures (Table 1). Using the best-performing layer to compute the average accuracy yields at least a 2% improvement. Similar findings were shown in (Fan et al., 2024) for generation tasks, while our results are for embedding tasks.

4.2 Downstream Performance is Negatively Correlated with Entropy

We evaluate the relationship between the prompt entropy of different layers with the performance on downstream tasks in the Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) dataset. The MMLU is designed to assess the comprehensive knowledge and multitask accuracy of language models. It comprises a wide range of subjects organized into 57 tasks, covering topics from elementary mathematics to professional law.

We compared two models of the same parameter size, Llama3-8B and Mamba2-8B. Despite having the same parameter size, Llama3 $63.85 \pm 0.38\%$ outperforms Mamba2 26.76 ± 0.37 significantly. We hypothesize that LLama3's superior performance is due to compression in its intermediate layers, as shown in Figure 1, enabling it to filter irrelevant information, which is useful for tasks like MMLU. Additionally, we find a strong negative correlation (-0.43) between the second and later layers of LLama3's representations and MMLU task performances 5. In contrast, Mamba2 shows neither such compression nor correlation with performance 6.

4.3 Experimental Setup for Quantifying Representation Quality

Next, we apply the metrics described in Section 3.3 to measure the quality of layerwise representations. We conduct experiments on Transformers, SSMs and Pythia (Biderman et al., 2023) using models of varying sizes to analyze the impact of architecture and capacity. We utilize the WikiText-103 dataset (Merity et al., 2016) and an instruction medical dataset (Vsevolodovna, 2024) to test different input complexities.

4.3.1 Differences Between Architectures

Our analysis reveals key differences in representation quality between Transformer-based architectures such as Pythia and SSMs such as Mamba across multiple metrics, including entropy, InfoNCE, LIDAR, and DiME. Figure 1 illustrates how these metrics vary as a function of model depth, represented as a percentage of the total number of layers, allowing for fair comparison between models of different depths.

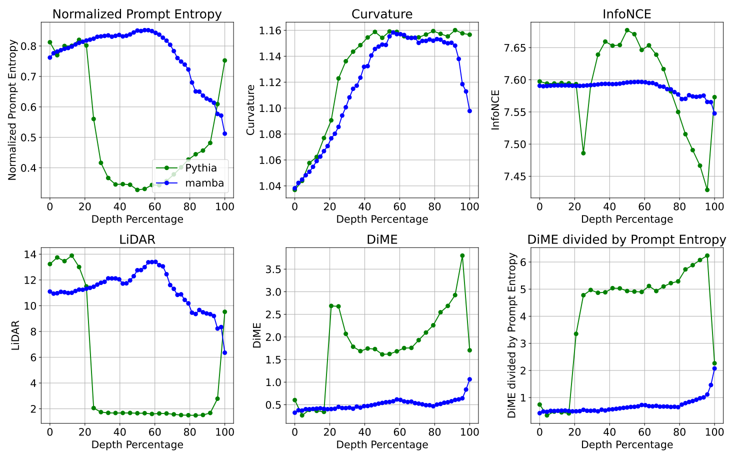


Figure 1: **Intermediate layers in Mamba show more stable representation values than Pythia, which exhibits more pronounced changes.** Representation evaluation metrics across layers in Pythia 410M and Mamba 370M architectures. The x-axis is the depth percentage of the model to allow fair comparison between models with different numbers of layers.

For entropy and LIDAR metrics, Pythia shows a significant reduction in values at intermediate layers, suggesting compression and information consolidation, while Mamba maintains more stable values, indicating less compression in intermediate representations. In contrast, Mamba exhibits lower values for the DiME and InfoNCE metrics than Pythia, implying less variability in intermediate representations.

The effect and changes in these metrics across the intermediate layers are generally less pronounced in Mamba than in Pythia. This indicates that Mamba maintains more stable representations throughout its depth, whereas Pythia exhibits greater shifts and transformations in its intermediate representations, potentially leading to different strengths in how these models encode and utilize information for downstream tasks.

4.3.2 Impact of Training Progression

To understand how representation quality evolves during training, we use the training checkpoints provided by Pythia, examining how the metrics change across different layers as training progresses. Figure 2 shows the evaluation metrics for logarithmically spaced training checkpoints, from the initial step to the final step at 143k.

The training dynamics reveal that the most significant changes occur in the intermediate layers. Specifically, the prompt entropy decreases in the middle layers during training, suggesting that the model learns to better compress and abstract the information within a prompt. This compression indicates that the model is becoming more efficient in representing complex information as training progresses. In contrast, the InfoNCE metric peaks in the middle layers, indicating increased distinctiveness of representations, while the LiDAR and DiME metrics both decrease, suggesting reduced variability in certain directions of the representation space.

Interestingly, the metrics in the initial layers remain relatively stable throughout the training, which we believe supports the detokenization hypothesis discussed in (Lad et al., 2024). This indicates that the initial layers primarily focus on mapping input tokens to an initial embedding space, with little change in their representation dynamics during training.

4.3.3 Prompt Entropy under Extreme Input Conditions

To further understand how prompt entropy behaves under different input conditions, we examine the effect of increasingly extreme prompts on the model's representations. Specifically, we analyze how the prompt entropy changes across layers of the Pythia 410M model when the input prompts exhibit high levels of token repetition, randomness, or increased length.

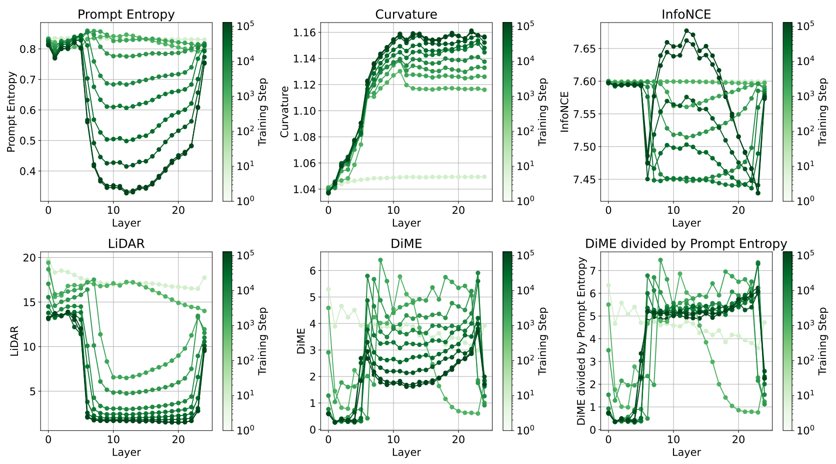


Figure 2: **Training effects are most pronounced in the intermediate layers, with distinct dynamics for different metrics.** Representation evaluation metrics across layers at various training checkpoints, ranging from step 1 to the final step at 143k. The x-axis represents the depth percentage of the model, showing how training affects different layers, particularly in the intermediate stages.

We design three types of extreme prompts:

1. **Prompts with Increasing Token Repetition**: We take 1000 regular prompts from the WikiText dataset and randomly replace tokens with a fixed token from the prompt at varying probabilities p. As p increases, the amount of repetition in the prompt increases.
2. **Prompts with Increasing Token Randomness**: We randomly replace tokens in the prompts with random tokens from the vocabulary at varying probabilities p. This introduces increasing levels of randomness into the prompts.
3. **Random Prompts of Increasing Length**: We generate random prompts by sampling tokens uniformly from the vocabulary, creating prompts of varying lengths T.

Figure 3 illustrates how the normalized and unnormalized prompt entropy changes across layers for these extreme prompts. Our key findings are as follows:

(1) Increasing token repetition leads to a decrease in entropy in the intermediate layers. As repetition increases, the model compresses redundant information, resulting in lower entropy values in the middle layers. This indicates that the model effectively identifies and encodes repetitive patterns.

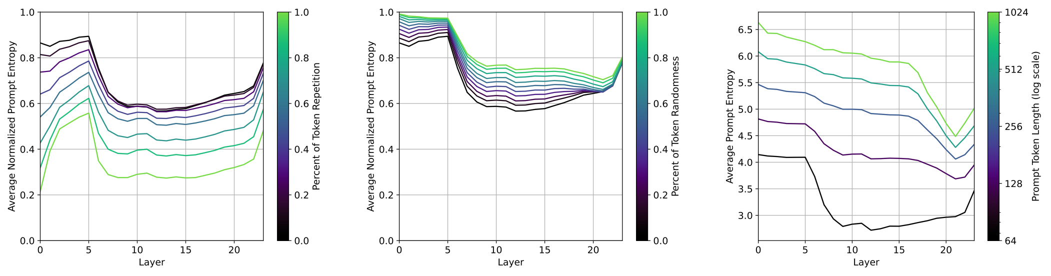
(2) Increasing token randomness results in higher entropy, especially in initial layers. Introducing random tokens increases the diversity of token representations, leading to higher entropy. The initial layers are more affected, suggesting sensitivity to input noise.

(3) Prompt length affects entropy in both normalized and unnormalized forms. Unnormalized entropy naturally increases with prompt length due to more tokens. While not displayed, the normalized entropy shows sublinear growth, indicating that each additional token contributes less to the overall diversity as the prompt becomes longer.

These observations highlight how extreme input conditions impact the model's internal representations, particularly in the intermediate layers. The model exhibits different compression and encoding behaviors depending on the nature of the input perturbations, which provides valuable insight into its processing mechanisms.

4.4 Bimodal Behavior in Prompt Entropy

While analyzing the average prompt entropy across different layers, we discovered an intriguing phenomenon: a clear bimodal distribution in the entropy values at certain layers in transformer models, but not SSMs. Figure 4 shows the entropy distributions for WikiText and the ai-medical-chatbot datasets(Vsevolodovna, 2024). Notably, a pronounced bimodal distribution is observed in the



(a) Repetition (b) Randomness (c) Random Prompt Length

Figure 3: **Prompt entropy across layers of Pythia 410M under different extreme input conditions.** (a) Increasing token repetition leads to decreased entropy in intermediate layers. (b) Increasing token randomness results in higher entropy, especially in the initial layers. (c) Unnormalized prompt entropy of random prompts increases with prompt length due to the larger number of tokens. These results demonstrate how the model's internal representations adapt to different types of input perturbations.

middle layers for the ai-medical-chatbot dataset. This behavior suggests that the model processes some prompts fundamentally differently than others at these intermediate stages. We investigated the causes of this behavior in Appendix A and ruled out prompt length, semantic complexity, or overlap with training data. The underlying cause is currently an open question.

5 Discussion and Conclusion

In this study, we thoroughly examined the quality of layerwise representations in LLMs, specifically comparing Transformer-based architectures and SSMs. Using a variety of evaluation metrics, including prompt entropy, curvature, InfoNCE, LIDAR, and DiME, we found several key insights into how these models process and encode information across different layers and under various conditions.

Our findings indicate that intermediate layers consistently provide superior representations for downstream tasks compared to final layers. This highlights the importance of using intermediate representations for feature extraction and transfer learning applications. Additionally, significant architectural differences were observed: Transformers exhibited more dynamic changes in metrics such as entropy and InfoNCE in their intermediate layers, suggesting a higher degree of information compression and variability. In contrast, SSMs maintained more stable representations, reflecting a different approach to information encoding that emphasizes consistency.

During training progression, the most substantial changes in representation quality occurred in the intermediate layers, with prompt entropy decreasing and InfoNCE peaking, indicating enhanced compression and distinctiveness of representations. This underscores the critical role of intermediate layers in the learning process and suggests potential avenues for optimizing training strategies to further improve representation quality.

LLMs demonstrated distinct behaviors under extreme input conditions, such as increased token repetition, randomness, and prompt length. Transformers showed significant variations in entropy and other metrics in response to input perturbations, particularly in intermediate layers, whereas SSMs maintained more stable representations. This suggests that transformers are more adaptable and sensitive to diverse input scenarios, while SSMs offer greater robustness and consistency. A particularly intriguing observation was the presence of bimodal entropy distributions in the intermediate layers, especially within the ai-medical-chatbot dataset. Despite extensive investigations, the cause of this bimodality remains unresolved.

In conclusion, our research advances the understanding of internal representation dynamics in LLMs, highlighting the pivotal role of intermediate layers and the distinct behaviors of different architectures. These findings not only contribute to the theoretical knowledge of model representations, but also offer practical guidance for optimizing model design, training, and application. Future work should delve deeper into the causes of phenomena such as bimodal entropy distributions and explore the development of new metrics tailored specifically to LLMs to further enhance representation evaluation.

References

Kumar K Agrawal, Arnab Kumar Mondal, Arna Ghosh, and Blake Richards. α-ReQ: Assessing representation quality in self-supervised learning by measuring eigenspectrum decay. *Advances in Neural Information Processing Systems*, 35:17626–17638, 2022.

Guillaume Alain. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Francis Bach. Information theory with kernel methods. *IEEE Transactions on Information Theory*, 69(2):752–775, 2022.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*, 2024.

Ido Ben-Shaul, Ravid Shwartz-Ziv, Tomer Galanti, Shai Dekel, and Yann LeCun. Reverse engineering self-supervised learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 58324–58345. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/b63ad8c24354b0e5bcb7aea16490beab-Paper-Conference.pdf`.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Paul Boes, Jens Eisert, Rodrigo Gallego, Markus P Müller, and Henrik Wilming. Von neumann entropy from unitarity. *Physical review letters*, 122(21):210402, 2019.

Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent. Guillotine regularization: Why removing layers is needed to improve generalization in self-supervised learning. *arXiv preprint arXiv:2206.13378*, 2022.

Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. *arXiv preprint arXiv:2309.07311*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *International conference on machine learning*, pp. 10929–10974. PMLR, 2023.

Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C Principe. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 61(1):535–548, 2014.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

John A Hartigan and Pamela M Hartigan. The dip test of unimodality. *The annals of Statistics*, pp. 70–84, 1985.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Eghbal Hosseini and Evelina Fedorenko. Large language models implicitly learn to straighten neural sentence trajectories to construct a predictive representation of natural language. *Advances in Neural Information Processing Systems*, 36, 2024.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. Exploring concept depth: How large language models acquire knowledge at different layers? *arXiv preprint arXiv:2404.07066*, 2024.

Vedang Lad, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*, 2019.

Xing Han Lù. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring, 2024. URL `https://arxiv.org/abs/2407.03618`.

Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL `https://arxiv.org/abs/2210.07316`.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In *International Conference on Learning Representations (ICLR)*, 2018.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2018.

Ravid Shwartz-Ziv. Information flow in deep neural networks. *arXiv preprint arXiv:2202.06749*, 2022.

Ravid Shwartz Ziv and Yann LeCun. To compress or not to compress—self-supervised learning and information theory: A review. *Entropy*, 26(3):252, 2024.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Ravid Shwartz-Ziv, Randall Balestriero, Kenji Kawaguchi, Tim GJ Rudner, and Yann LeCun. An information theory perspective on variance-invariance-covariance regularization. *Advances in Neural Information Processing Systems*, 36:33965–33998, 2023.

Oscar Skean, Jhoan Keider Hoyos Osorio, Austin J Brockmeier, and Luis Gonzalo Sanchez Giraldo. DiME: Maximizing mutual information by a difference of matrix-based entropies. *arXiv preprint arXiv:2301.08164*, 2023.

Oscar Skean, Aayush Dhakal, Nathan Jacobs, and Luis Gonzalo Sanchez Giraldo. FroSSL: Frobenius norm minimization for self-supervised learning. In *European Conference on Computer Vision*, 2024.

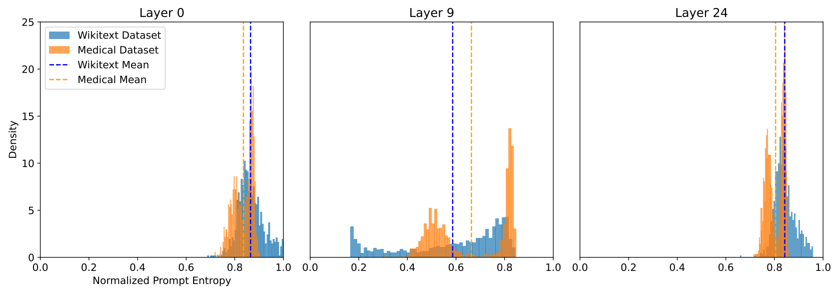
Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M Susskind, and Etai Littwin. LiDAR: Sensing linear probing performance in joint embedding ssl architectures. *arXiv preprint arXiv:2312.04000*, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

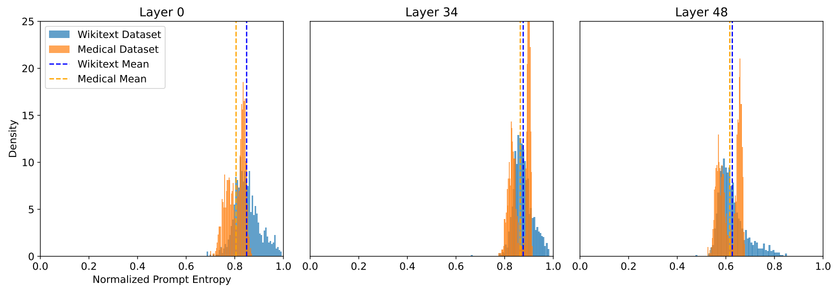
Ruslan Magana Vsevolodovna. Ai medical chatbot dataset, 2024. URL `https://huggingface.co/datasets/ruslanmv/ai-medical-chatbot`.

Lai Wei, Zhiquan Tan, Chenghai Li, Jindong Wang, and Weiran Huang. Large language model evaluation via matrix entropy. *arXiv preprint arXiv:2401.17139*, 2024.

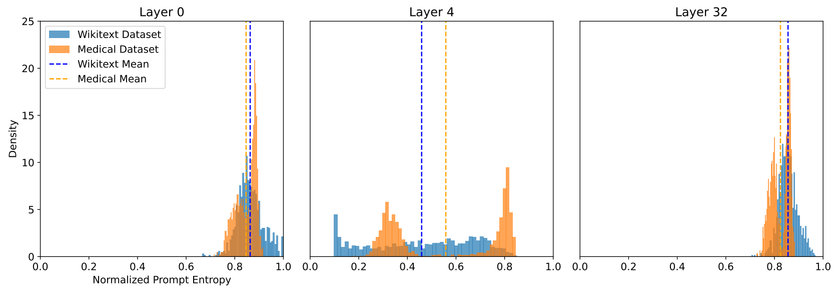
Zhanghao Zhouyin and Ding Liu. Understanding neural networks with logarithm determinant entropy estimator. *arXiv preprint arXiv:2105.03705*, 2021.



(a) Pythia 410M



(b) Mamba 370M



(c) Llama3 8B

Figure 4: **Bimodal distribution of prompt entropies observed in intermediate layers.** The distributions of prompt entropies for WikiText and ai-medical-chatbot datasets are shown for Pythia, Mamba, and Llama3 models. The middle column highlights the layer with the highest Dip Test score (Hartigan & Hartigan, 1985), which measures the degree of multimodality in the entropy distribution.

A Investigation into Bimodal Distribution of Entropies

To determine the underlying cause of this bimodal distribution of prompt entropies, we conducted several experiments to see if specific properties of the dataset could explain this phenomenon. Our goal was to understand whether the bimodality was related to characteristics such as prompt length, semantic complexity, or overlap with training data.

Effect of Prompt Length Initially, we hypothesized that the bimodality might be caused by variations in prompt length. If one mode corresponded to shorter prompts and the other to longer prompts, it could indicate different processing strategies. However, since the entropy values were normalized and theoretically invariant to length, this was unlikely. Upon further analysis, we confirmed that prompt length did not significantly correlate with the observed bimodality.

Manual Examination of Prompts We then manually examined prompts from each mode of the distribution to identify any distinguishing features, such as difficulty or specific types of medical terminology. Despite this effort, we found no significant differences between the prompts in either mode. Both modes contained a similar range of medical complexity and varied use of terminology, suggesting that the model's entropy was not merely a reflection of the difficulty or specificity of the input.

Training Set Overlap Next, we investigated whether the low entropy mode might be associated with prompts that were very similar to samples seen during training. Given that both the ai-medical-chatbot dataset and PILE (Gao et al., 2020) (which Mamba, Pythia, and possibly Llama3 were trained on) contained medical articles from PubMed, we hypothesized that overlap with training data could lead to more confident, lower-entropy representations. To test this, we implemented a BM25 index (Lù, 2024) to quickly search for identical or highly similar articles between the two datasets.

While we did find identical articles between the ai-medical-chatbot dataset and PILE, these articles were evenly distributed across both modes of the bimodal entropy distribution. This suggests that the presence of training set overlap does not explain the bimodal behavior, and the underlying cause remains an open question.

B Representation Evaluation Metrics

B.1 Token Embedding Diversity Metrics

Token embedding diversity metrics evaluate the variability and richness of the representations at the token level within a single sequence. These metrics are designed to capture how distinctively each token is represented within the context of the entire prompt, providing insight into how effectively the model encodes information and differentiates between different parts of the input.

Prompt Entropy Following Wei et al. (2024), we use the α-order matrix-based entropy (Giraldo et al., 2014) as a surrogate for Rényi entropy. For a sequence of token representations $\mathbf{Z} \in \mathbb{R}^{L \times d}$, the Gram matrix is $\mathbf{K_Z} = \mathbf{Z}\mathbf{Z}^\top$. The entropy is computed as:

$$S_\alpha(\mathbf{Z}) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{L} \left(\frac{\lambda_i(\mathbf{K_Z})}{\operatorname{tr}(\mathbf{K_Z})} \right)^\alpha \right). \tag{1}$$

In this context, prompt entropy measures the diversity and dispersion of token embeddings within a given sequence. Higher entropy values imply a richer and more varied representation of the tokens, suggesting that the model captures more nuanced information across the sequence. This helps in understanding how effectively the model encodes diverse features and maintains the complexity of the input, making it a useful metric for evaluating the quality of intermediate layer representations. Unless otherwise specified, we use the limit case of $\alpha = 1$ in our calculations. Details and behavior for different α are shown in Appendix D.

Curvature As introduced by Hosseini & Fedorenko (2024), curvature measures the change in direction between adjacent token embeddings. To calculate curvature, we first we calculate the difference between two adjacent vectors as $\mathbf{v}_k = \mathbf{z}_{k+1} - \mathbf{z}_k$. The average curvature of a prompt is:

$$\bar{C} = \frac{1}{L-2} \sum_{k=1}^{L-2} \arccos \left(\frac{\mathbf{v}_{k+1}^\top \mathbf{v}_k}{\|\mathbf{v}_{k+1}\| \|\mathbf{v}_k\|} \right). \tag{2}$$

B.2 Augmentation Invariance Metrics

These metrics assess the robustness of representations to input augmentations. Because augmentation may change the length of the tokenized prompt, the token embedding diversity metrics described in B.1 are no longer suitable. Instead, we average the tokens to get a single vector representing each prompt and use the metrics described below to measure the similarity between two augmentations of the same prompt. We refer to the two batches of augmented prompts as $Z_1 \in \mathbb{R}^{N \times D}$ and $Z_2 \in \mathbb{R}^{N \times D}$, where N is the batch size and row i in both matrices correspond to the same original prompt. We provide full details and examples of the augmentation process in Appendix F.

InfoNCE We compute a mutual information lower bound using the InfoNCE loss (Oord et al., 2018) between two views. This loss is widely used to train augmentation-invariant networks in self-supervised learning for vision and is well-suited to capturing the semantic similarity underlying the augmented prompts (Chen et al., 2020a,b; Shwartz Ziv & LeCun, 2024; Ben-Shaul et al., 2023).

DiME Similarly to infoNCE, the quantity DiME (Skean et al., 2023; Chen et al., 2023) can be used to measure a mutual information-like between the two augmented batches of prompts. DiME is based on the matrix-based entropy described in Eq. 1. Roughly put, DiME measures the quality of the pairings between Z_1 and Z_2 as compared to between Z_1 and ΠZ_2 for some permutation matrix Π. A low value of DiME means the row pairings between Z_1 and Z_2 are no better than random pairings.

LiDAR The LiDAR quantity (Thilak et al., 2023) was proposed to act as a representation quality metric. Unlike matrix-based entropy which looks at the principal component variances, LiDAR uses the linear discriminant component variances. To compute the linear discriminant analysis (LDA) matrix, LiDAR uses augmentations to construct the class scatter matrix. In our setting, we use N classes (each corresponding to different prompt) with J samples per class (each sample within a class being a different augmentation of the same prompt). Due to the more complex requirements of computing the LDA matrix, we use $J = 16$ rather than $J = 2$ like in DiME or infoNCE.

C Architectural Details

In this section, we elaborate on the specific architectures of Transformers and State Space Models (SSMs). We outline the mathematical foundations, including the weight matrices, attention mechanisms for Transformers, and the state transition matrices for SSMs. Detailed equations and parameter configurations are provided to facilitate replication and deeper understanding.

C.1 Transformer

The Transformer architecture (Vaswani, 2017) utilizes self-attention mechanisms. Given an input $\mathbf{x}$, the key ($\mathbf{K}$), query ($\mathbf{Q}$), and value ($\mathbf{V}$) matrices are computed as:

$$\mathbf{Q} = \mathbf{x}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{x}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{x}\mathbf{W}_V, \tag{3}$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$ are learned weights.

The attention weights are calculated using:

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right), \tag{4}$$

where $\mathbf{M}$ is a mask to enforce causality in autoregressive tasks.

The output is then:

$$\mathbf{y} = \mathbf{A}\mathbf{V}. \tag{5}$$

C.2 State Space Models

SSMs (Gu & Dao, 2023) model sequences using recurrent dynamics. The hidden state $\mathbf{h}_t$ and output $\mathbf{y}_t$ at time t are updated as:

$$\mathbf{h}_t = \mathbf{A}\mathbf{h}_{t-1} + \mathbf{B}\mathbf{x}_t, \tag{6}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t + \mathbf{D}\mathbf{x}_t, \tag{7}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times d}$, $\mathbf{C} \in \mathbb{R}^{d \times n}$, and $\mathbf{D} \in \mathbb{R}^{d \times d}$ are learned parameters.

D Behavior of Matrix-based Entropy for different choices of α

One way to interpret Eq. 1 is as the α-order Rényi entropy of the Gram matrix eigenvalues[1]. Notice how each eigenvalue is divided by $\text{tr}(\mathbf{K_Z})$ before being raised to the α power. This is so that the eigenvalues of $\mathbf{K_Z}$ sum to one (because $\text{tr}(\cdot) = \sum_{i=1}^{n} \lambda_i(\cdot)$), which is a necessary condition to treat the eigenvalues as a probability distribution. Furthermore, each eigenvalue of $\mathbf{K_Z}$ signifies the variance of samples in a particular principal component direction Scholkopf & Smola (2018). If entropy is low, then the eigenvalues form a heavy-tail distribution which implies that a few components dominate the variance of samples in Z. On the other hand, at maximum entropy, the eigenvalues form a uniform distribution and samples are spread equally in all directions. Matrix-based entropy is reminiscent of the LogDet entropy which uses the determinant of $\mathbf{K_Z}$ to capture how much "volume" a dataset occupies Shwartz-Ziv et al. (2023); Zhouyin & Liu (2021). The LogDet entropy is given by $S_{\text{LogDet}}(Z) = \log\det(\mathbf{K_Z}) - \log 2$. One can use Jensen's inequality to show that the LogDet entropy is a lower bound of Eq 1 when $\lim_{\alpha\to 1}$ (Appendix J.4 of Shwartz-Ziv et al. (2023)).

Depending on the choice of α, several special cases of matrix-based entropy can be recovered. In particular, when $\lim_{\alpha\to 1}$ it equals Shannon entropy (also referred to as von Neumann entropy in quantum information theory Bach (2022); Boes et al. (2019)), and when $\alpha = 2$ it equals collision entropy. Interestingly, the case of $\alpha = 2$ can be calculated without explicit eigendecomposition Skean et al. (2024). We show in the Appendix Figure 7 how varying values of α affects the matrix-based entropy of Gram matrices with eigenvalues distributed with a β-power law such that $\lambda_i = i^{-\beta}$. It is shown that for larger values of α, smaller eigenvalues contribute more to the entropy.

E Dataset Details

E.1 Wikitext Dataset

We used the wikitext dataset Merity et al. (2016) for the majority of our experiments in Section 4.3. This was downloaded from **Salesforce/wikitext** on huggingface. The dataset consists of 100 million tokens scraped from the Featured articles on wikipedia. We filtered out prompts which were less than 30 tokens or were wikipedia section headings.

E.2 AI-Medical-Chatbot Dataset

We also used the medical instruction dataset called ai-medical-chatbot Vsevolodovna (2024) which downloaded from **ruslanmv/ai-medical-dataset** on HuggingFace. An example from this dataset is:

```
You are an AI Medical Assistant Chatbot, trained to answer medical questions.
    Below is an instruction that describes a task, paired with an response
    context. Write a response that appropriately completes the request.

### Instruction:
What is the resurgent sodium current in mouse cerebellar Purkinje neurons?

### Context:
FGF14 modulates resurgent sodium current in mouse cerebellar Purkinje neurons.
```

F Prompt Augmentations

For the augmentation-invariance metrics such as infoNCE, LiDAR, and DiME, we use the NLPAug library Ma (2019) to augment our prompts. We use three types of augmentations.

- The SplitAug augmentation randomly splits words into two parts by adding a space.
- The RandomCharAug augmentation randomly inserts, substitutes, swaps, or deletes characters.

[1]The non-zero eigenvalues of the Gram matrix ZZ^T are equivalent to those of the covariance matrix Z^TZ. Using the covariance matrix instead of the Gram matrix in Eq. 1 makes no difference and is more computationally efficient if $D < N$.

- The Keyboard augmentation randomly substitutes characters with other characters that are at a distance of one as measured on a QWERTY keyboard. For instance, the character "k" may be replaced with "i", "l", "m", or "j".

We use the pseudocode below to do our augmentations using three types of augmentations, using the default library settings for each type. When computing augmentation-invariance metrics like infoNCE or DiME, we use the two augmented prompts rather than using one augmented prompt alongside the original prompt. Note that these augmentations may change the token length T of a prompt.

```
aug = naf.Sequential([
    naw.SplitAug(p=0.3),
    nac.RandomCharAug(p=0.3),
    nac.KeyboardAug(p=0.3),
])
(aug_A, aug_B) = aug.augment(prompt, num_augmentations=2)

prompt -> "The quick brown fox jumps over the lazy dog."

aug_A ->  "The quDUk b rown fox wEmps o ver the l azy dog."
aug_B ->  "The qTuXi bro wn fox uVm)s ob3r the la_k dog."
```

G Extreme Prompts

G.1 Increasing Repetition

We take regular prompts from the wikitext dataset, tokenize them, and then for each token we randomly replace it with probability p. We draw replacements tokens by sampling a random token from within the prompt. We show examples below for varying levels of p.

- ($p = 0$) Mint records indicate the first gold dollars were produced on May 7...
- ($p = 0.1$) Mint records indicate the first gold dollars were Mint Mint May 7...
- ($p = 0.5$) Mint records Mint Mint Mint gold dollars were Mint Mint Mint 7...
- ($p = 1.0$) Mint Mint Mint Mint Mint Mint Mint Mint Mint Mint Mint Mint Mint...

G.2 Increasing Randomness

We take regular prompts from the wikitext dataset, tokenize them, and then for each token we randomly replace it with probability p. We draw replacements uniformly from the tokenizer distribution. We show examples below for varying levels of p. Unlike the character-level random noise added to prompts in Section with random noise discussed in Appendix F which might change the number of tokens T of the prompt, the token-level random noise used here does not do so.

- ($p = 0$) Mint records indicate the first gold dollars were produced on May 7...
- ($p = 0.1$) Mint records indicate salivary first gold dollars were produced on May NaCl...
- ($p = 0.5$) Mint records Dallas actively first dollars persufors on Mayder129 18...
- ($p = 1.0$) arf emulsion minorensteinorianmega_TOStack potsRecip Installifykeeping...

G.3 Random Prompts with Certain Length

To make a random prompt of a specific length T, we sample T tokens uniformly from the Pythia tokenizer distribution. Such a prompt may look like the following for $T = 16$: "Proposition Sequencespecific Exp fibers brows Club overviewNos toss Thinking traderMulti indoorlis".

We show how random prompt representations evolve over Pythia training checkpoints in Figure 8. The random prompts we use are of length 512 tokens. It is readily observed that the prompt entropy is flat across layers in the beginning of training. As training progresses, the model compresses more and more near the final layers.

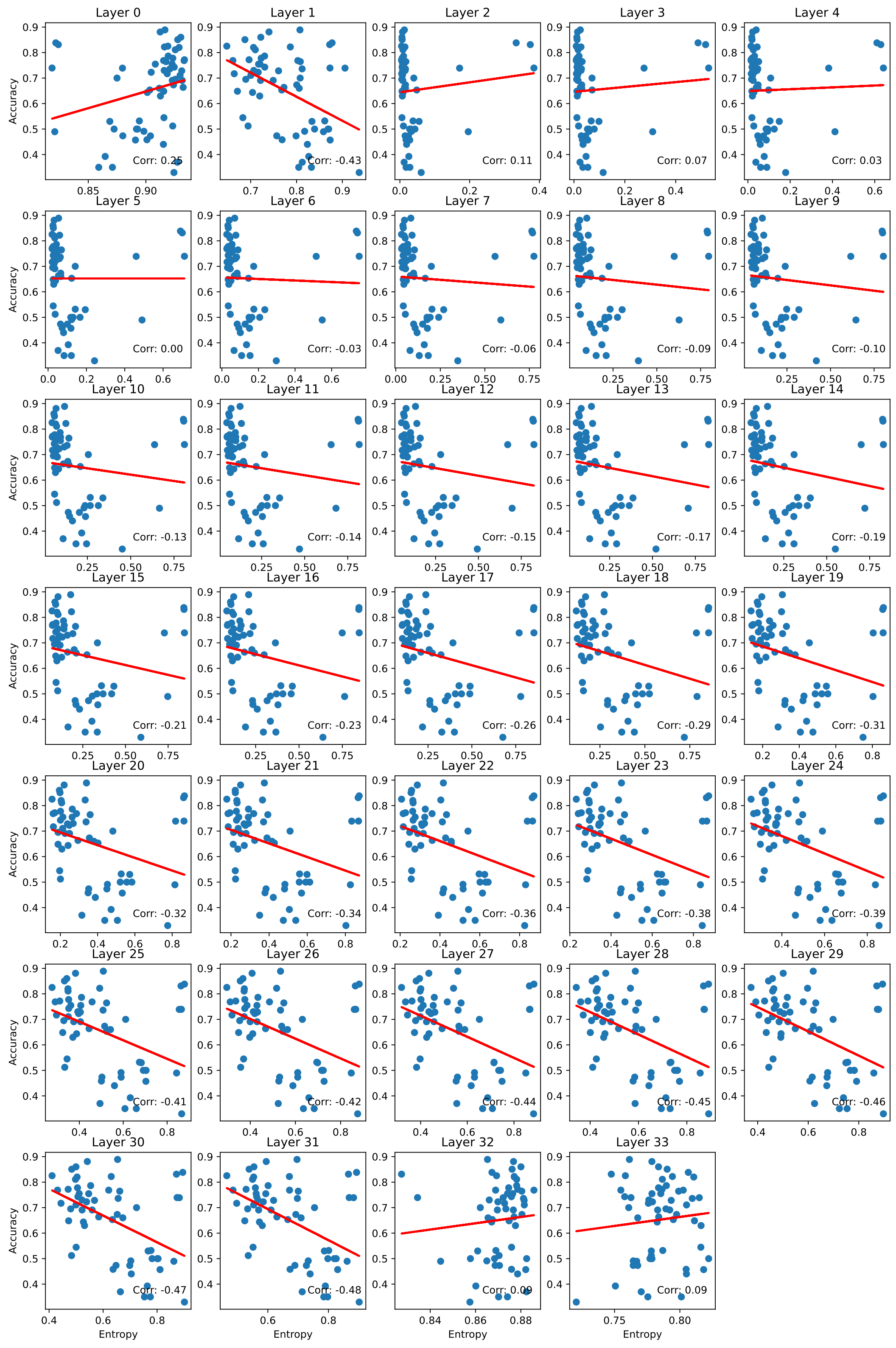


Figure 5: Entropy vs Accuracy of LLama3-8B on MMLU tasks. Each point represents a task in MMLU

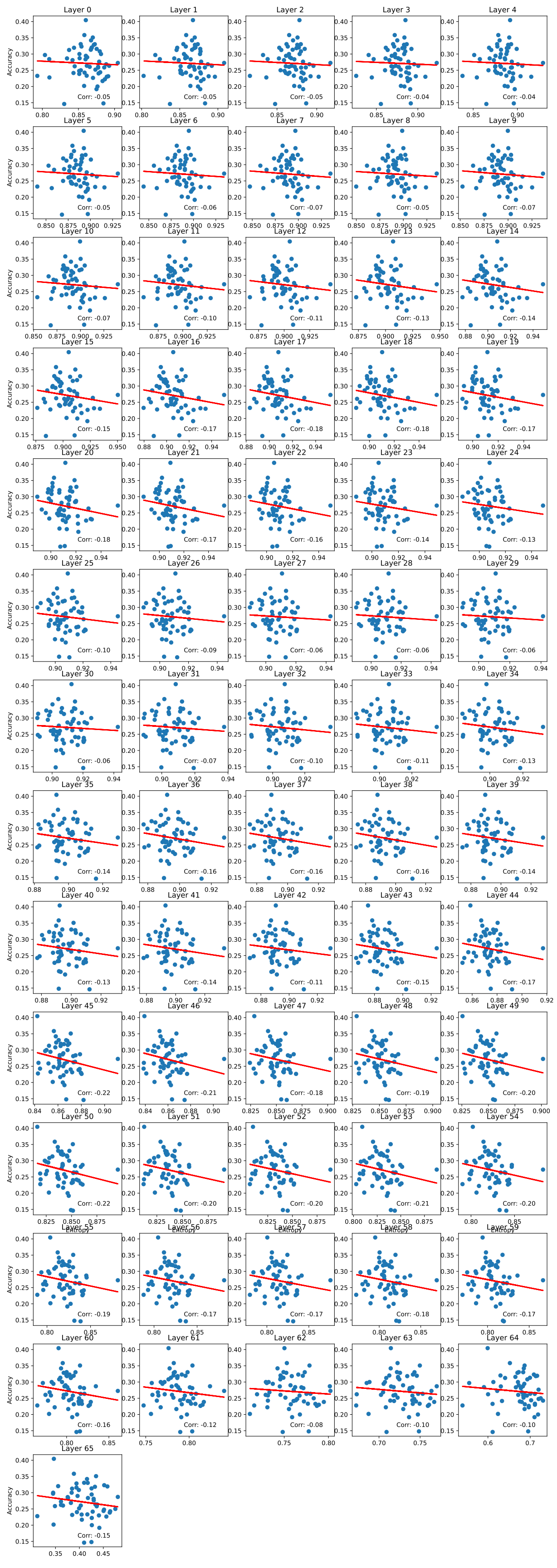


Figure 6: Entropy vs Accuracy of Mamba2-8B on MMLU tasks

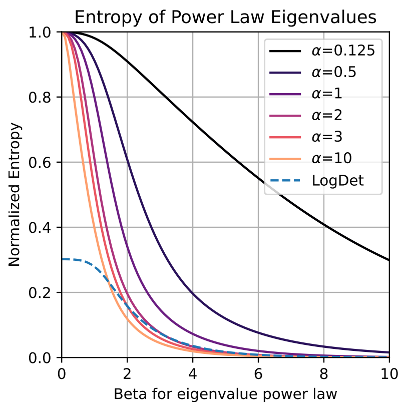


Figure 7: The behavior of Eq. 1 for varying values of α on Gram matrices with eigenvalues distributed with a β-power law such that $\lambda_i = i^{-\beta}$.

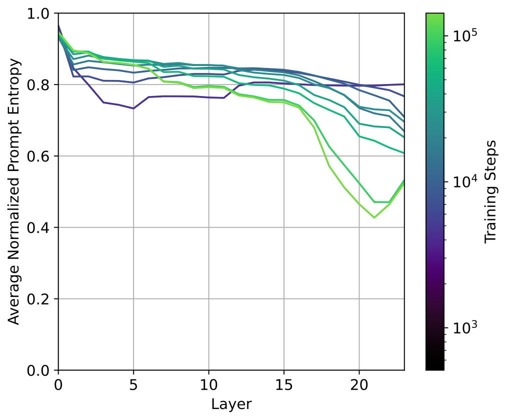


Figure 8: Behavior of random prompt representations as model is training