VISUALIZING LLM LATENT SPACE GEOMETRY THROUGH DIMENSIONALITY REDUCTION

Alex Ning[*]
Department of Computer Science
University of Virginia
rnx2bc@virginia.edu

Vainateya Rangaraju[*]
Department of Computer Science
University of Virginia
prr3gw@virginia.edu

Yen-Ling Kuo
Department of Computer Science
University of Virginia
ylkuo@virginia.edu

ABSTRACT

Large language models (LLMs) achieve state-of-the-art results across many natural language tasks, but their internal mechanisms remain difficult to interpret. In this work, we extract, process, and visualize latent state geometries in Transformer-based language models through dimensionality reduction. We capture layerwise activations at multiple points within Transformer blocks and enable systematic analysis through Principal Component Analysis (PCA) and Uniform Manifold Approximation (UMAP). We demonstrate experiments on GPT-2 and LLaMa models, where we uncover interesting geometric patterns in latent space. Notably, we identify a clear separation between attention and MLP component outputs across intermediate layers, a pattern not documented in prior work to our knowledge. We also characterize the high norm of latent states at the initial sequence position and visualize the layerwise evolution of latent states. Additionally, we demonstrate the high-dimensional helical structure of GPT-2's positional embeddings, the sequence-wise geometric patterns in LLaMa, and experiment with repeating token sequences. We aim to support systematic analysis of Transformer internals with the goal of enabling further reproducible interpretability research. We make our code available at `https://github.com/Vainateya/Feature_Geometry_Visualization`.

1 Introduction

Despite enormous advances made in the field of machine learning (ML) research, understanding a model's internal decision-making processes remains a difficult challenge. Model interpretability is central to areas such as alignment and explainable AI, and the field is burgeoning with an influx of work. However, many foundational questions are still unresolved. Against this backdrop, mechanistic interpretability has emerged as a field that aims to achieve a granular, causal understanding of neural networks, particularly large language models (LLMs), by reverse engineering their internal components. One promising avenue for understanding LLMs is analyzing their representations. Feature geometry, the structure and organization of representations within high-dimensional latent space, offers a way to study how abstract features are encoded and transformed across model layers. By examining geometric relationships between latent states, such as directions, clusters, and manifolds, researchers can gain insight into how LLMs generalize, reason, and abstract.

In this work, we analyze the feature geometry of LLMs by capturing latent states across multiple components and projecting them into interpretable low-dimensional spaces using PCA and UMAP. This approach allows visualizations of how representations evolve through the Transformer model. Throughout this work, we aim to provide a strong background and exposition for better accessibility. To ensure clarity, a glossary of key technical terms used throughout this paper is provided in section 5.

[*]Equal contribution

2 Background

2.1 Transformers

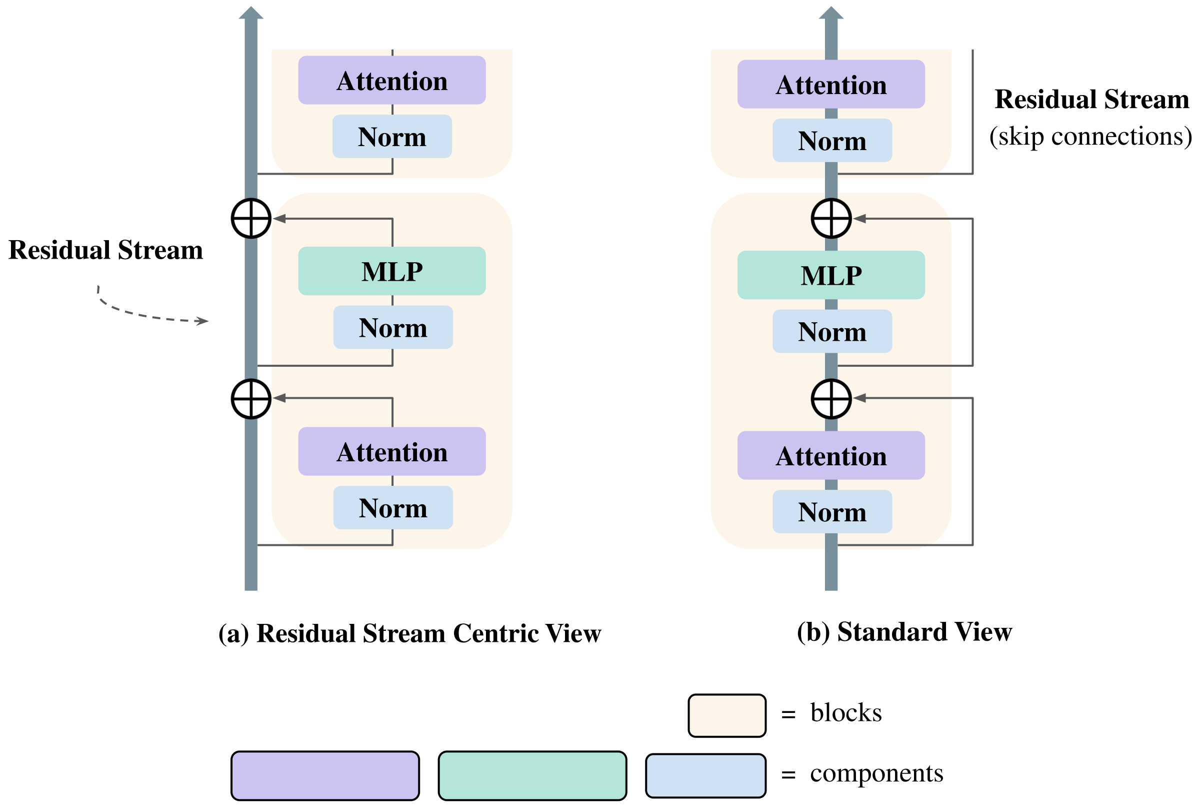

Figure 1: The two equivalent perspectives on the Transformer architecture

Originally introduced by Vaswani et al. in 2017 [1], Transformers have achieved great notoriety for their state-of-the-art performance across most language modeling tasks. Despite this success, their internal mechanisms remain difficult to understand, motivating extensive work in mechanistic interpretability (see Section 1). Since our experiments focus on analyzing decoder-only Transformers (e.g., GPT-2 [2] and LLaMa [3]), we begin by providing a brief overview of the Transformer architecture. To ensure clarity amongst various terminologies in recent literature, we also establish a consistent set of terms that will be used throughout this paper. That is, we formalize a decoder-only Transformer as a sequence of **blocks**, each consisting of four primary **components/layers**: the normalization layer (e.g., layer norm or RMSNorm) preceding the attention component, the multi-head self-attention, the normalization layer preceding the MLP, and the MLP (multilayer perceptron). Although the original Transformer architecture utilized a post-norm design, where the normalization layer came after the attention or MLP component, most modern architectures (GPT-2 and LLaMa) adopt a pre-norm design. Thus, we depict the pre-norm architecture in figures 1 and 3. We use the term "final norm" to refer to the normalization applied after the last Transformer block. This is immediately followed by the unembedding layer, which maps the final latent representation into vocabulary logits for token prediction. Finally, we use 0-based indexing when referring to blocks, layers, and sequence positions throughout the paper. For example, "block 0" denotes the *first* block in the Transformer, and "sequence position 0" denotes the *first* token position in a sequence. In order to avoid ambiguity between "first" and index "1", we will refer to the "first" (0-th) item as the "initial" item. While Transformers are often conceptualized as a sequential operation through each of their components with residual connections between layers, we adopt the mathematically equivalent perspective that these skip connections collectively form the central communication channel through which all components interact. We refer to this pathway as the residual stream. [4].

Figure 1 illustrates these differing perspectives. In this residual stream-centric view, all components of a Transformer (the token embedding, attention heads, MLP layers, and unembedding layer) communicate with each other by reading and writing to different subspaces of the residual stream. Rather than thinking of information flowing sequentially through layers, we conceptualize each layer as reading its input from the residual stream (by performing a linear projection), and then writing its result to the residual stream by adding a linear projection back in [4]. This additive structure preserves a path for an identity pathway to flow through the model and into each component, unobstructed by any operation other than the direct sum of component updates back into the residual stream. We highlight the importance of the residual stream perspective for two reasons. Firstly, it allows us to treat all components of the Transformer as operating within a shared representational space, intuitively describing the collaborative nature of component interactions. This shared space enables different components to develop a collective understanding of features and their corresponding directions, making it possible to interpret representational changes as coordinated rather than isolated transformations. Secondly, this framing provides a conceptual basis for analyzing how the geometry of these shared representations evolves throughout the residual stream, thereby offering insights into how components jointly shape the model's feature space.

2.2 Linear Representation Hypothesis

LLMs learn internal features, which are abstract properties or characteristics of data that guide their predictions and generalization. In language modeling, such features often correspond to linguistic attributes such as gender, tense, or sentiment. Mikolov et al. [5] showed that word embeddings capture these features through simple vector arithmetic. For example, $\langle \text{King} \rangle - \langle \text{Man} \rangle + \langle \text{Woman} \rangle \approx \langle \text{Queen} \rangle$, where angle brackets denote embedding vectors. This provided the first evidence that semantic and syntactic relations can be represented as linear patterns in latent space. This empirical finding has since been formalized in the context of large language models as the Linear Representation Hypothesis (LRH) [6], which posits that high-level concepts are represented as directions (i.e., one-dimensional subspaces) in representation space. This implies an intuition where learned representations are approximately linear combinations of feature directions.

LRH is significant for mechanistic interpretability in two respects. First, LRH implies a property of decomposability, which conveniently states that complex representations can be analyzed in terms of independent, concept-aligned features. This property provides a tractable foundation for understanding LLM latent space dynamics. Second, under the assumption that features are represented linearly, dimensionality reduction methods can more reliably reveal the organization and combination of features within the latent space. In this framing, each feature i may be represented by a unit vector e_i, with an input's representation expressed as $\sum x_i e_i$, where x_i denotes the feature's strength or presence [7].What that said, linear representation hypothesis remains a *hypothesis*, and there are limitations to the interpretations it can provide. Transformer models implement many behaviors which are extremely complex, not yet understood, and highly non-linear. [8] explore features which are *not* one-dimensionally linear. Additionally, the Superposition Hypothesis [7], posits that in order to represent a large amount of independent features in a space with limited dimensionality, neural networks permit a certain amount of interference between features by embedding them into *almost*-orthogonal directions instead of fully orthogonal directions. Although still linear, this degree of interference potentially clouds the benefits of interpretability and decomposability offered by LRH.

2.3 Dimensionality Reduction

Modern neural networks generate latent representations that live in extremely high-dimensional spaces. While these representations contain rich structural information, they are difficult to interpret directly. Dimensionality reduction techniques serve as a bridge between high-dimensional representations and human visualization. The goal is to compress embeddings into lower dimensions while preserving the salient structure, e.g., variance, clusters, or manifold geometry, so that latent state behavior can be more easily examined. In this work, we study both Principal Component Analysis (PCA) and Uniform Manifold Approximation (UMAP) as the primary methods for dimensionality reduction and visualization.

PCA is a widely used method for linear dimensionality reduction. PCA operates by finding a new orthogonal basis that captures directions of maximal variance in the data [9]. By projecting embeddings onto these principal axes, PCA allows one to rank-order components by their explanatory power. The method is computationally efficient, analytically grounded in linear algebra through eigenvalue decomposition or singular value decomposition, and does not require hyperparameter tuning. However, PCA is limited to capturing linear relationships and assumes that variance corresponds to meaningful structure, an assumption that may fail in non-Gaussian or highly nonlinear datasets [9]. Despite these caveats, PCA remains a powerful baseline, is widely used in our experiments, and provides a stable reference frame for comparing geometric patterns.

On the other hand, Uniform Manifold Approximation and Projection (UMAP) [10], provides a nonlinear alternative designed to preserve both local neighborhoods and some aspects of global manifold structure. Rather than identifying variance-maximizing axes, UMAP builds a graph-based representation of data as a topological manifold and then optimizes a low-dimensional embedding that preserves this structure. In practice, UMAP is effective at revealing clustering patterns in latent states, producing interpretable visualizations. We chose UMAP over other nonlinear alternatives such as t-SNE [11] because it better maintains global relationships while still revealing local clustering patterns. Its flexibility comes at the cost of tunable hyperparameters (e.g., number of neighbors, minimum distance) and a lack of linear structure, but it has become a widely adopted tool in machine learning for exploring latent structure.

2.4 Positional Embeddings/Encodings

Transformers are inherently permutation-invariant, meaning that without explicit positional information, they cannot distinguish between tokens based solely on order. To address this, models incorporate positional information into token representations so that attention mechanisms can account for sequence structure. This information is typically introduced in one of two ways: through learned positional embeddings or functional positional encodings, each with distinct implications for how visualizations depict positional geometry in a model's latent space.

In many early Transformer architectures, such as GPT-2 (used in our experiments), positional information is injected via learned positional embeddings. Each token position in the input sequence is associated with a learned vector that is added to its corresponding token embedding before entering the first Transformer block. Because these embeddings are learned during training, they often exhibit consistent geometric patterns that reflect the model's internal representation of positional order. Within the context of our work, learned embeddings provide a straightforward entry point for feature geometry analysis, serving as a baseline for how position-dependent features evolve through the residual stream.

On the other hand, the LLaMa model we use employs a specific form of functional positional encoding called Rotary Positional Encodings (RoPE). Rather than adding fixed position vectors, RoPE applies a position-dependent rotation to query and key vectors within each attention head. This rotation encodes relative position information directly into the attention mechanism, allowing the model to generalize more effectively to longer sequences. Conceptually, RoPE embeds position in the geometry of relationships between tokens, rather than as an explicit positional vector.

This distinction serves as the key motivation behind *Effects of Sequence Position* experiments in section 4.5 and *Repeating Token Experiments* in section 4.6.

3 Methodology

The visualizations obtained from our dimensionality reduction experiments are the results of a 3-part process designed to capture, organize, and explore the internal representations of Transformer models. Figure 2 shows the overview of our methodology. The first component is the pipeline, which handles the extraction of latent states by running models on text data and saving layerwise latent states alongside metadata and token information. Once this latent state dataset has been generated, the second component, data processing, provides tools for efficiently loading, filtering, and manipulating the stored representations. Finally, the third component focuses on dimensionality reduction and visualization, utilizing PCA and UMAP to project high-dimensional latent states into interpretable views. Together, these stages provide a unified workflow for moving from raw model activations to structured visualizations that highlight the geometric signatures present in the latent space.

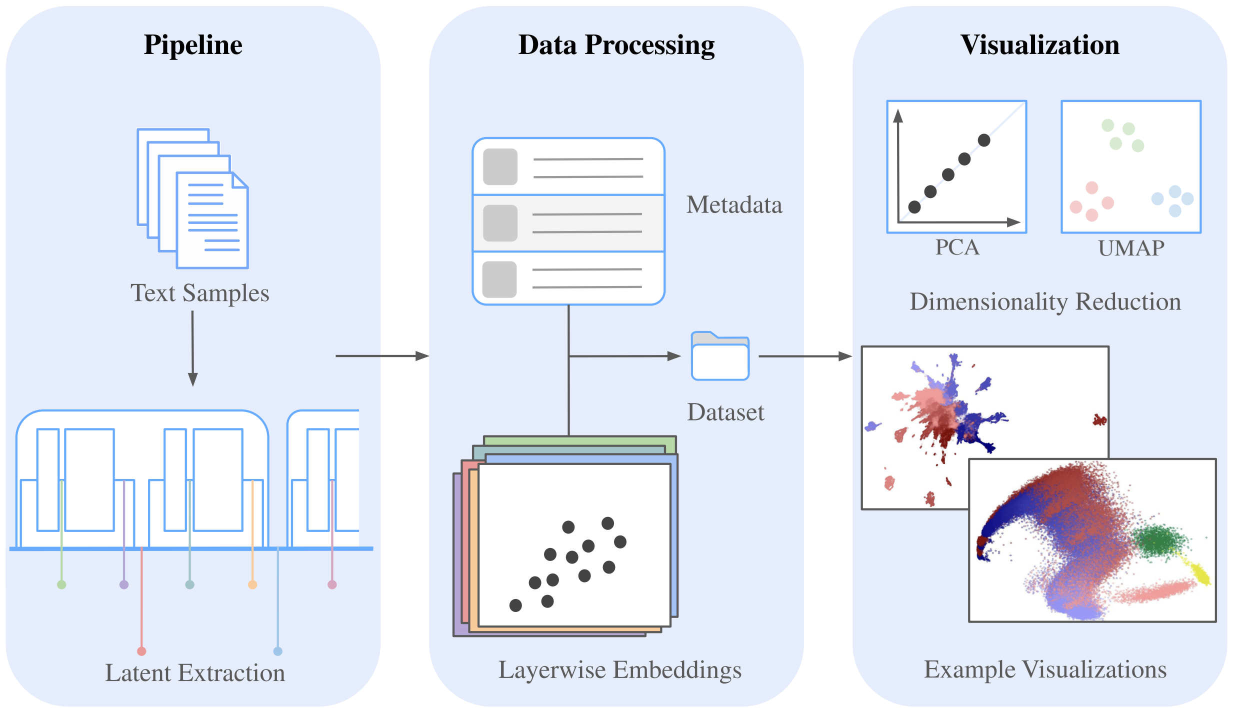

Figure 2: Overview of visualization pipeline: Text samples first pass through Transformer layers for latent extraction, these latent states are then organized into a structured dataset alongside metadata, then reduced via dimensionality reduction for interpretable visualizations.

3.1 Pipeline

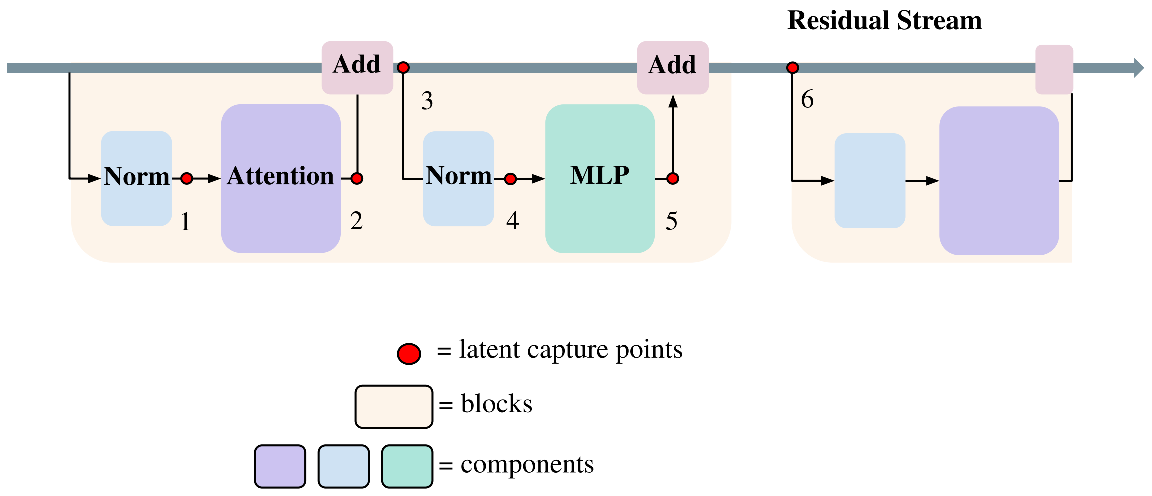

Figure 3: The six capture points within each Transformer block. Points 1 and 4 correspond to the outputs of the normalization layers (pre-attention and pre-MLP). Points 2 and 5 correspond to the outputs of the attention and MLP modules, respectively. Points 3 and 6 capture the residual stream after the attention and MLP additions.

The generation pipeline automates three stages: (1) generating inputs, (2) capturing latent states, and (3) saving labeled latent states for downstream processing and visualization. This design provides a reproducible and extensible foundation for our experiments (see section 4).

The pipeline supports two modes of operation. In the default "text" mode, text passages are sampled from a text dataset. For our experiments, we ran the model through the well-known Project Gutenberg (PG-19) dataset [12], consisting of a corpus of books written before 1919. Each passage is tokenized and truncated or padded to a fixed sequence length, ensuring uniformity across samples. In "singular" mode, by contrast, the pipeline probes individual tokens directly by iterating over the model's vocabulary. This allows for fine-grained analysis of token-level representations without contextual interference. Both modes are parameterized by the number of samples and the desired sequence length (set at 1 for the singular mode).

Figure 3 shows each of the capture points along the Transformer architecture. For each block, we record the outputs of each layer: both normalizations, the attention module, and the MLP. We denote the raw outputs of the attention and MLP components and their preceding norms as "pre-add" (points 1, 2, 4, and 5 in Figure 3). From these activations, we also compute the residual stream after each addition (post-attention and post-MLP), capturing the points where new computations are integrated into the residual pathway. We denote these as "post-add" (points 3, 6 in Figure 3). In total, this yields six distinct captures per Transformer block.

Captured latent states of high dimensionality can optionally undergo post-processing to moderately reduce their dimensionality via PCA (e.g., from 4096 to 512 dimensions). This reduced dimensionality can substantially reduce memory usage and loading, processing, and visualization time with minimal impact on results.

3.2 Dimensionality Reduction and Visualization

Captured latent states are visualized in 2D following dimensionality reduction. Not all captured latent states must be used for dimensionality reduction and visualization. Instead, a subset may be selected (e.g., only latent states from particular layers). For dimensionality reduction, both PCA and UMAP are supported, with a GPU-accelerated implementation for UMAP (via cuML [13]) utilized for visualizing large amounts of latent states. Before dimensionality reduction is performed, augmentations to the latent states can be applied. These include:

- Converting latent states to unit length, which emphasizes directional structure over norm.
- Averaging the latent state vectors over any dimension(s) (sample dimension, sequence dimension, layer dimension). This may be done to reduce the variance of a certain dimension to concentrate on the structure present in another.

PCA transform is performed without mean centering to preserve the position of the origin.

We utilize a customizable color-coding scheme to distinguish the layer and sequence dimensions. In our experiments, attention layers are blue while MLP layers are red. Darker shades indicate earlier layers while lighter shades indicate later ones. For the sequence dimension, we use a gradient to indicate earlier vs. later positions.

Fitted dimensionality reduction models can be reused; PCA fitted on one set of latent states can be applied to a different set of latent states, and visualizations can preserve x- and y-axis limits between runs. This is especially useful for ablation studies, where subtle differences between experimental conditions must be compared in a consistent coordinate frame. Additionally, dimensionality reduction can be performed, which reduces the number of dimensions to a number *greater* than 2. Then, any pair of the reduced dimensions can be visualized together in 2D (e.g., dimensions 2 and 5 or 3 and 7). Figure 11 shows the result of visualizing all possible pairs of dimensions after dimensionality reduction to 6 dimensions (totaling $\binom{6}{2} = 15$ pairs), then stitching the resulting visualizations into a grid.

4 Results and Discussion

4.1 Experimental Setup

We used GPT-2 Large (774M) and LLaMa-7B to generate the latent states. On all results using the PG-19 dataset, we use the following input shapes:

- GPT-2: 128 samples, each 1024 tokens long
- LLaMa: 64 samples, each 2048 tokens long

We note that we use the maximum input length possible for both models. LLaMa is trained with a `<BOS>` (beginning of sentence) token prepended to all inputs. As a result, we prepend the `<BOS>` token to the beginning of all PG-19 inputs to LLaMa. After collecting the LLaMa latent states, we reduced the dimensionality of the latent states from 4096 to 512 using PCA. These dimensionality-reduced latent states were subsequently used in all visualizations except those in section 4.6. They were not used in any analyses of latent state norms; those used the original full dimensionality latent states for accurate results. All GPT-2 latent states were used in their full dimensionality.

Existing research demonstrates how different blocks in a Transformer have different roles. While earlier blocks focus on converting tokens into concepts [14] and the final blocks heavily denoise and refine the output, intermediate (middle) blocks, which are more robust to swapping and deletion, gradually create and refine intermediate features in a shared representation space [15, 16]. For some of our experiments, we are interested in investigating the behavior of the intermediate block latent states without interference from earlier or later blocks. As a result, we define here the specific "intermediate" blocks/layers which we will later refer to in our results and discussions for both our tested models. Because there does not exist a clear methodology to define concrete boundaries for the "intermediate" blocks, the definitions we make are somewhat arbitrary. Nevertheless, it is important for us to have consistent definitions across our analyses. For GPT-2, we define these as the layers in blocks 2-8 (from a range of 0-11), while for LLaMa, they are the layers in blocks 6-27 (from a range of 0-31).

4.2 Large Norm of 0-th Sequence Position Latent States

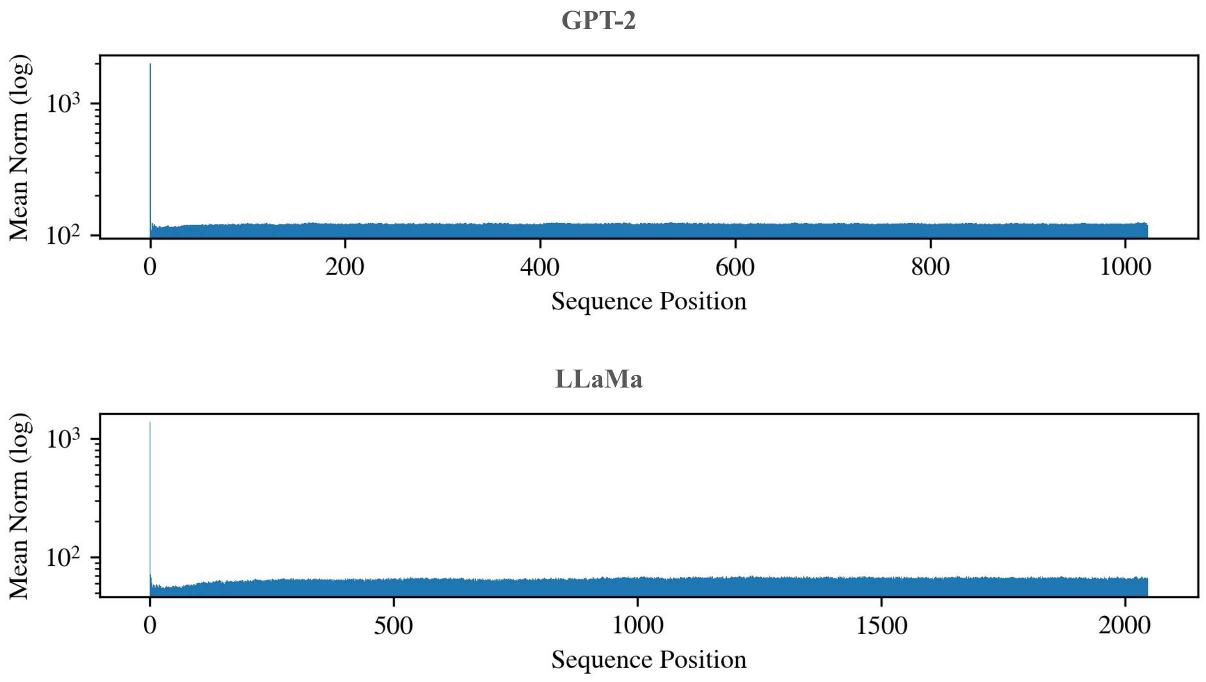

Figure 4: Norm of latent states from intermediate layers from both GPT-2 and LLaMa along sequence positions. Norms were averaged over both samples and layers. The dataset used is PG-19.

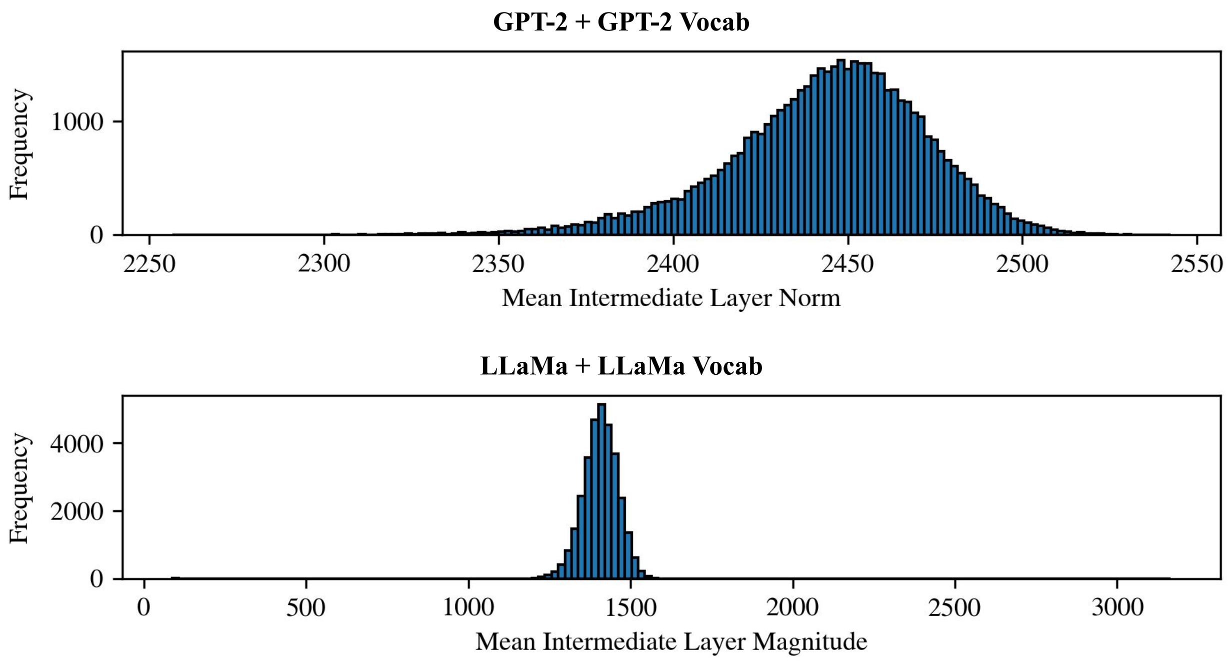

Figure 5: Histograms of the mean intermediate layer latent state norm from each vocab token for both GPT-2 and LLaMa. Each vocab token was input individually into each model as the initial token.

When performing dimensionality reduction using methods such as PCA or UMAP on a set of vectors (such as latent states), a few vectors with outlier norms can obscure the structures present between all other vectors in the set by overshadowing them[2]. We analyze the intermediate layer norms of PG-19 hidden states averaged over the sample and layer dimensions for each sequence position. Figure 4 shows large spikes in the norm of the 0-th sequence position token for both LLaMa and GPT-2. This phenomenon has been noticed and analyzed in prior works and has functions which include acting as a bias term [17].

As was similarly described by [17], we find that these large spikes in latent norm occur in the LLaMa model at the initial token of a sequence. We find this interesting, as although the LLaMa model uses a `<BOS>` token at the start of all inputs [3, 18], the spike in latent state norm for the 0-th sequence position is not limited to the appearance of the `<BOS>` token. Figure 5 shows histograms of the mean norm across intermediate layers for each vocab token in both GPT-2 and LLaMa when given as input to the model as the initial token. As can be seen, the vast majority of LLaMa vocab tokens take on large latent state norms when they are the initial token. We find this property intriguing as, unlike the learned positional embeddings of GPT-2, the *relative* RoPE positional encodings used by LLaMa do not provide the model with as trivial a way to determine whether an embedding belongs to a token in the 0-th sequence position. Nonetheless, tokens in the 0-th sequence position typically have a corresponding latent state in LLaMa with very high norm, indicating that LLaMa has likely developed a more complex mechanism to detect tokens at the 0-th sequence position.

[2]In the UMAP case, this is applicable when the chosen metric is sensitive to norm outliers, such as with the Euclidean distance metric. Other metrics, such as cosine distance, would not be affected by norm outliers.

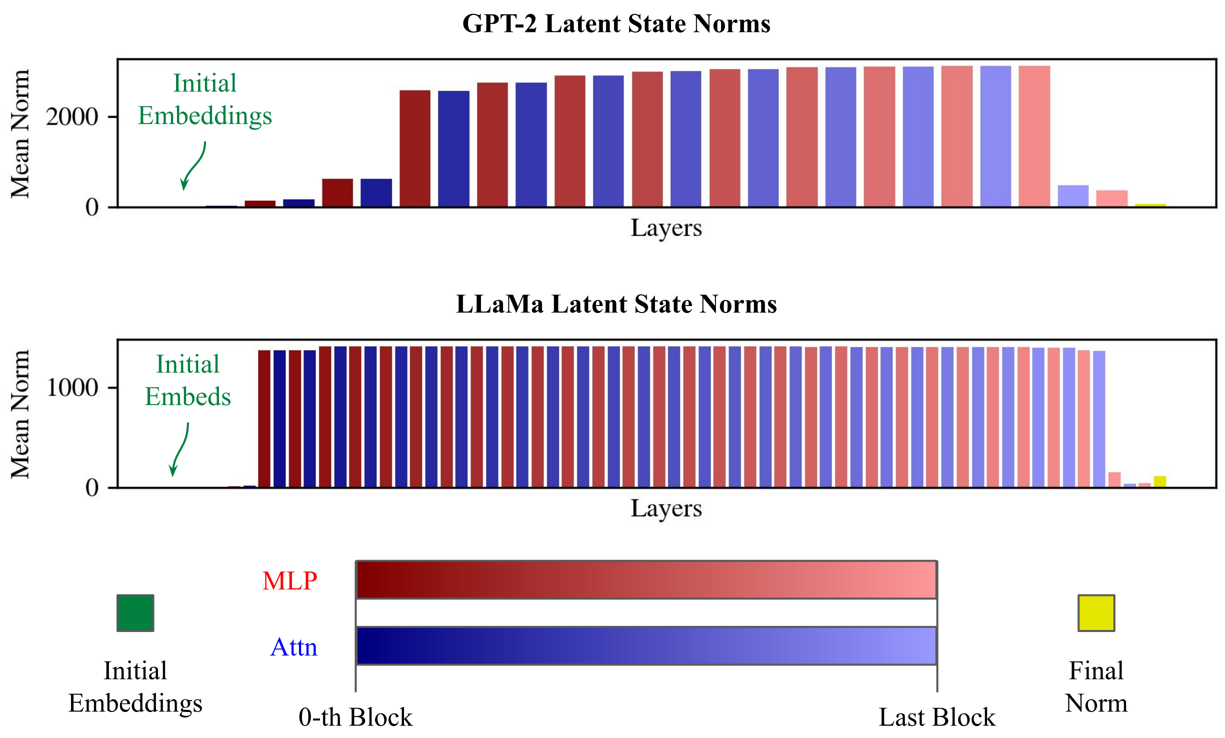

Figure 6: Latent state norms across each layer of both GPT-2 and LLaMa for initial token inputs. For each model, all of its vocab tokens were individually input into the model as the initial token, and the mean norm for each layer was taken over all the vocab tokens.

As was also described by [17], the high-norm latent states emerge after a few initial Transformer layers and then reduce in norm in the final layers. Figure 6 shows the mean norm across all individual vocab tokens (separately input into the model) for each layer of both GPT-2 and LLaMa. As can be seen, the layers in the middle have very high norms, while those at the beginning and end do not.

Due to this observation of high latent state norms in the 0-th sequence position, in subsequent results, we often omit the latent states of the 0-th sequence position when performing dimensionality reduction and visualization so that it does not obscure all details in other latent states.

4.3 Layerwise Visualizations

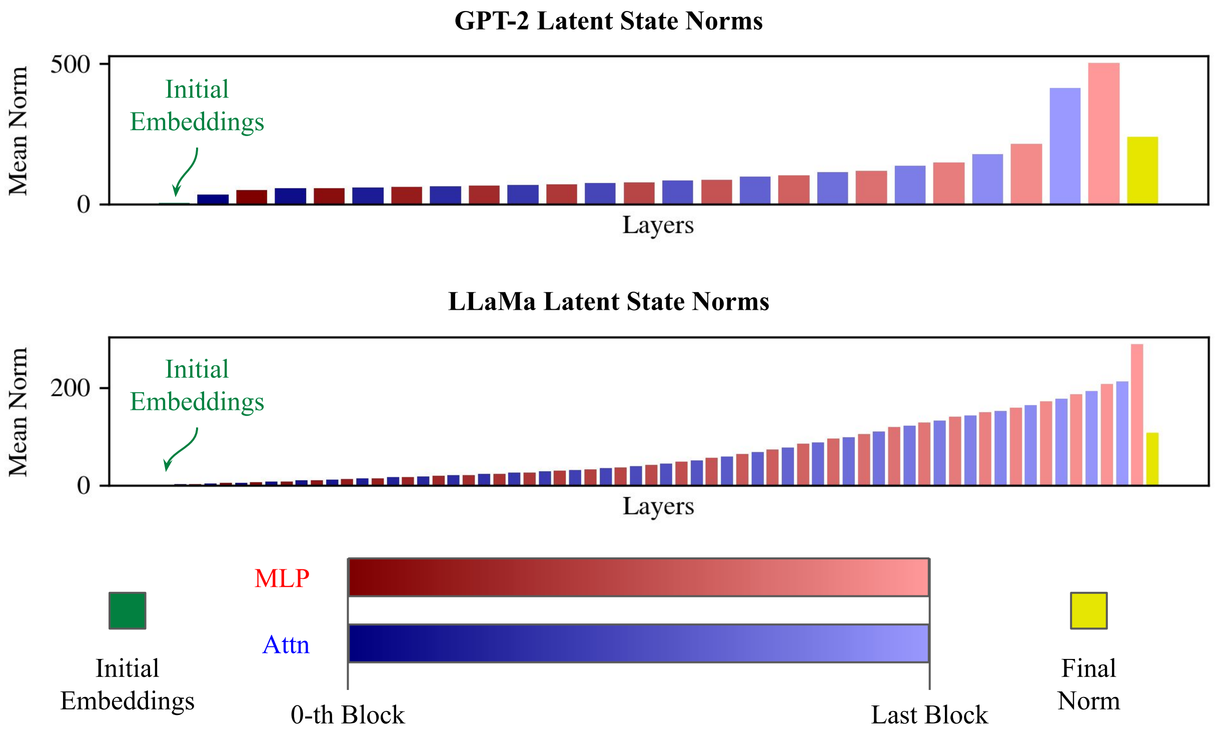

Figure 7: Latent state norms across each layer of both GPT-2 and LLaMa. The norms for each layer were averaged across all samples and sequence positions. Initial token latent states were excluded.

Figure 7 shows the average norm of each layer of both GPT-2 and LLaMa across sequences of the PG-19 dataset. Initial token hidden states are excluded. In contrast with the consistently large norms seen in the initial token latent states of Figure 6, the norms of Figure 7 gradually increase across the layers and are never as large. Furthermore, there are patterns which are consistent across both GPT-2 and LLaMa, such as the large spike in norm in the last block of the Transformer and the drop in norm after the final norm layer.

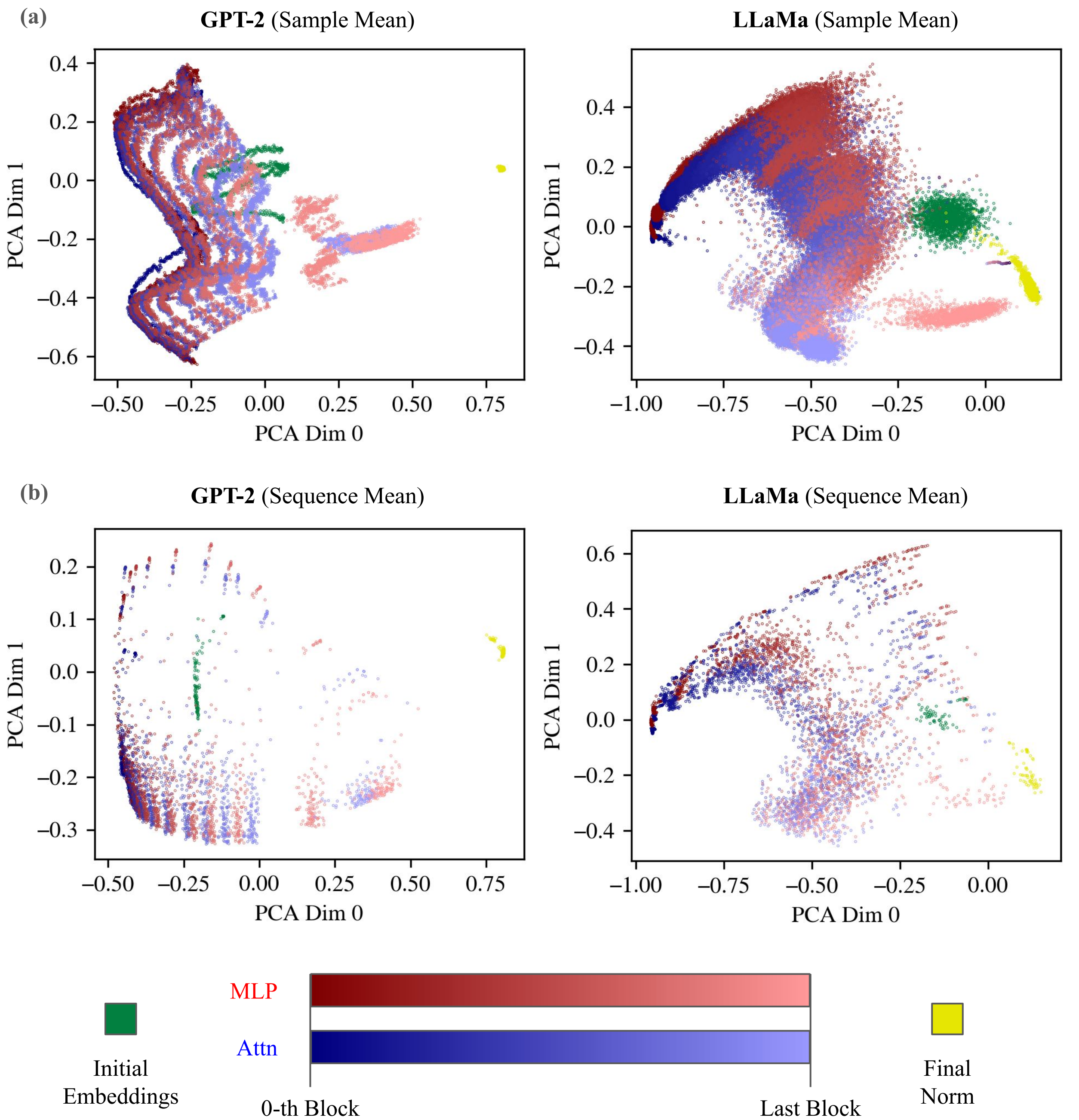

Figure 8: PCA visualizations of GPT-2 and LLaMa latent states on the PG-19 dataset. Latent states were converted to unit vectors before any PCA was performed. PCA was fit to all latent states, but was used to transform latent states only after they were averaged across samples/sequence dimensions. The initial token latent states were excluded. (a) Visualization after averaging across the sample dimension. (b) Visualization after averaging across the sequence dimension.

Figure 8 visualizes, using PCA, the layers of both GPT-2 and LLaMa in latent space. All latent states were converted to unit vectors before any dimensionality reduction or visualization was performed. Otherwise, the higher norms of the later layers tend to overpower all other variability, resulting in an uninteresting visualization. Additionally, while PCA was fit on the full latent state data, it was used to transform the data only after either averaging the latent states over the sequence or sample dimensions in order to reduce visual clutter (otherwise, there can be so many data points that the visualization is not intelligible). Averaging the latent states over the sample dimension removes the variability between samples, leaving the only sources of variability in the latent states being the differences between sequence positions and the differences between layers of the model. In contrast, averaging over the sequence dimension removes any variability/structure associated with the sequence position, thus only leaving the random variability between samples and the differences between layers. As can be seen in figure 8, a clear layer-wise progression of latent states is visible in both GPT-2 and LLaMa cases. In the GPT-2 case, the "wavy" structure of the positional embeddings is clearly visible in each layer of the sample mean (a) case, while in LLaMa, which uses RoPE, a pattern is less clear. The patterns of sequence position are further analyzed in section 4.5.

4.4 Attention vs. MLP Signature

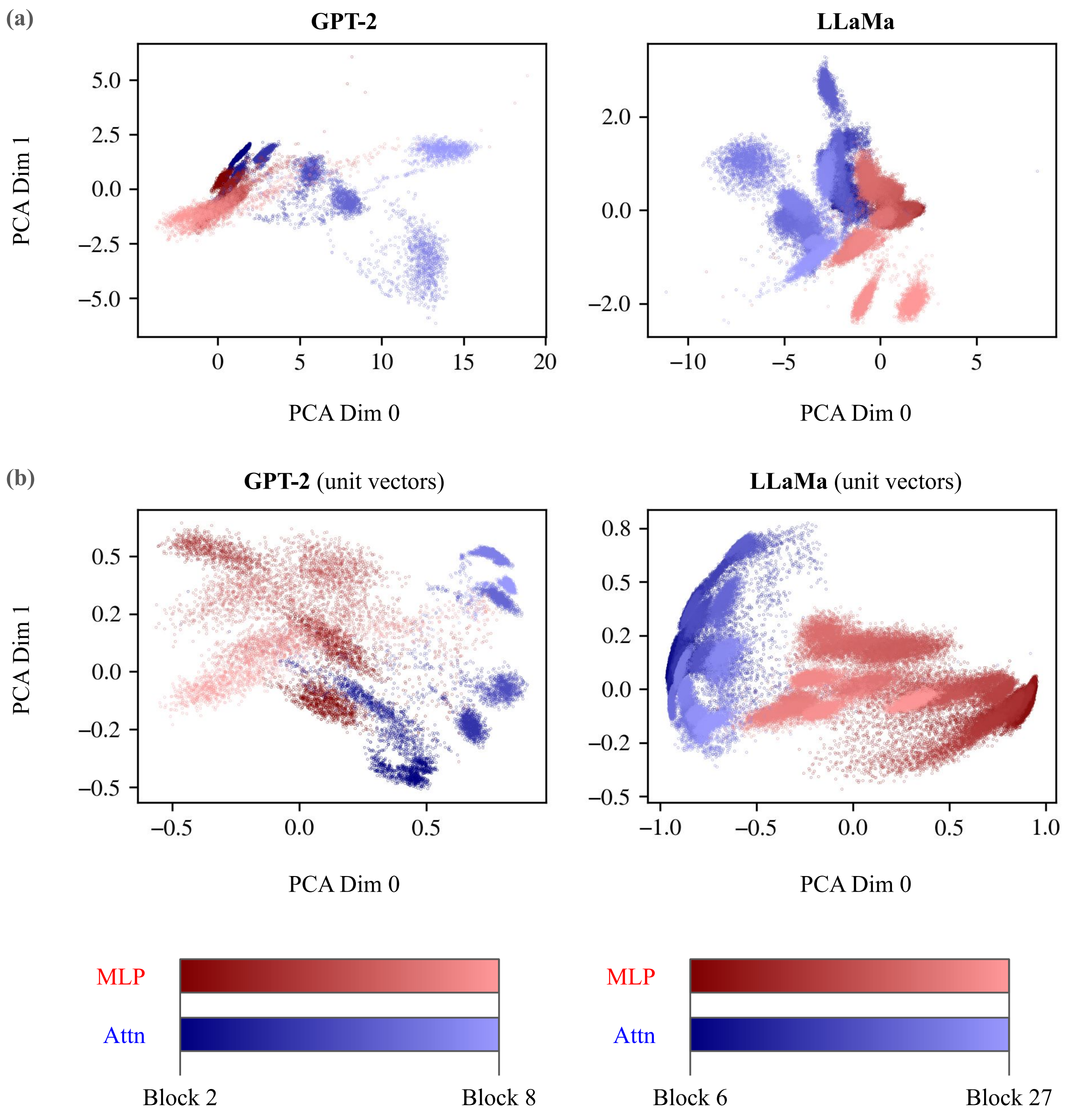

Figure 9: PCA visualizations of *pre-add* GPT-2 and LLaMa intermediate layer latent states on the PG-19 dataset. PCA was fit to all latent states but was used to transform latent states after they were averaged across samples. Initial token latent states were excluded. (a) PCA visualizations of GPT-2 and LLaMa intermediate block latent states (b) PCA visualizations of GPT-2 and LLaMa intermediate block latent states after being converted to unit vectors; unit vector data was used for both the fit and transform/visualize stages.

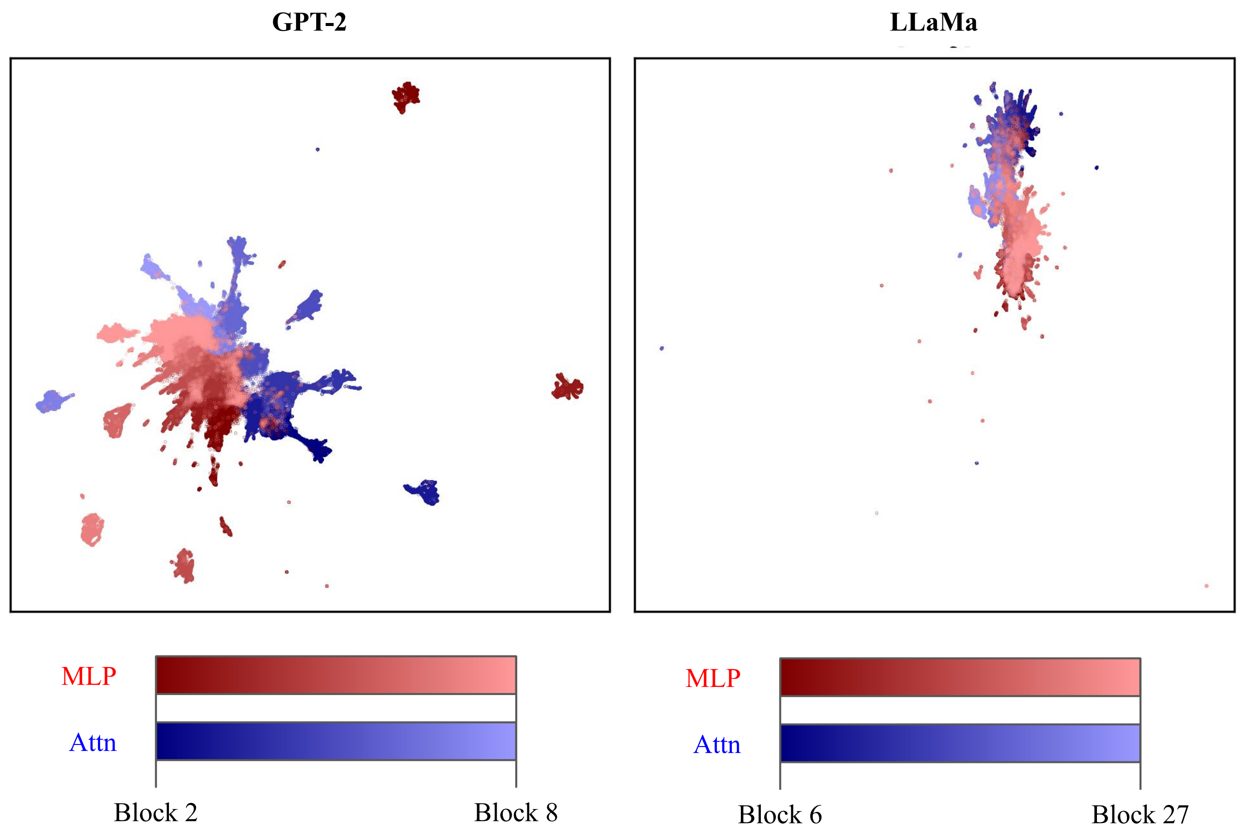

Figure 10: UMAP visualizations of GPT-2 and LLaMa *pre-add* intermediate layer latent states on the PG-19 dataset. For consistency with Figure 9, initial token hidden states are excluded.

We notice a striking separation between pre-add latent states from attention components when compared to latent states from MLPs in their dimensionality-reduced visualizations. That is, through many different Transformer blocks, the output of attention components tends to occupy a distinct area in the visualized space when compared to the output of MLPs. To the best of our knowledge, this has not yet been shown in previous studies.

Figures 9 and 10 visualize pre-add contributions from both attention components and MLPs in intermediate blocks using samples from PG-19. Figure 9 utilizes PCA dimensionality reduction on the latent states. Although the PCA dimensionality reductions in all subplots of Figure 9 were *fit* on all latent states, the actual *transformation* using the fitted PCA was done on the latent states only after they had all been averaged over the sample dimension to reduce random variability between samples for a clearer visualization. Figure 9(a) shows the PCA visualizations of both GPT-2 and LLaMa. (b) shows the PCA visualizations of both GPT-2 and LLaMa when all latent states were converted to unit vectors before PCA dimensionality reduction. In both (a) and (b), there is a clear separation of regions occupied by latent states from attention components (blue) versus MLPs (red).

Figure 10 utilized 2D UMAP dimensionality reduction on the latent states. Unlike in the PCA case of Figure 9, there was no sample averaging of the latent states. Instead, due to the increased computational demands of UMAP (using all latent states is not viable), a random subset of 100 k latent states was used to fit UMAP, while a *different* random subset of 500 k latent states was transformed and visualized. Similar to as seen in Figure 9, we can see distinct regions occupied by latent states from attention components (blue) compared to those from MLPs (red). The linear representation hypothesis specifies that features correspond to *directions* in latent space. As a result, we use cosine distance as the metric for UMAP instead of Euclidean distance.

4.5 Effects of Sequence Position

We visualize the geometric effects of sequence position on post-add latent states from intermediate blocks of both GPT-2 and LLaMa. PCA was used for dimensionality reduction instead of UMAP, as UMAP does not meaningfully preserve geometric shapes in Euclidean space. We use intermediate layer latent states from PG-19. Latent states were averaged over both samples and layers before dimensionality reduction. This was to eliminate sources of variability between samples or layers, leaving only the structure between sequence positions. Latent states were also converted to unit vectors before dimensionality reduction. This was done for two reasons. First, we found the visualizations looked either nearly identical (GPT-2) or slightly clearer (LLaMa) when latent states were converted to unit vectors. Second, the use of unit vectors allowed us to include the initial token latent state without being concerned that its extreme norm would obscure all other patterns. We provide visualizations that omit the initial token and do not convert latent states to unit vectors as a comparison in appendix 6.1. For all visualizations, we perform PCA which reduces the latent states to 6 dimensions and then visualize the 15 unique pairs of those 6 dimensions.

4.5.1 GPT-2 Positional Embeddings

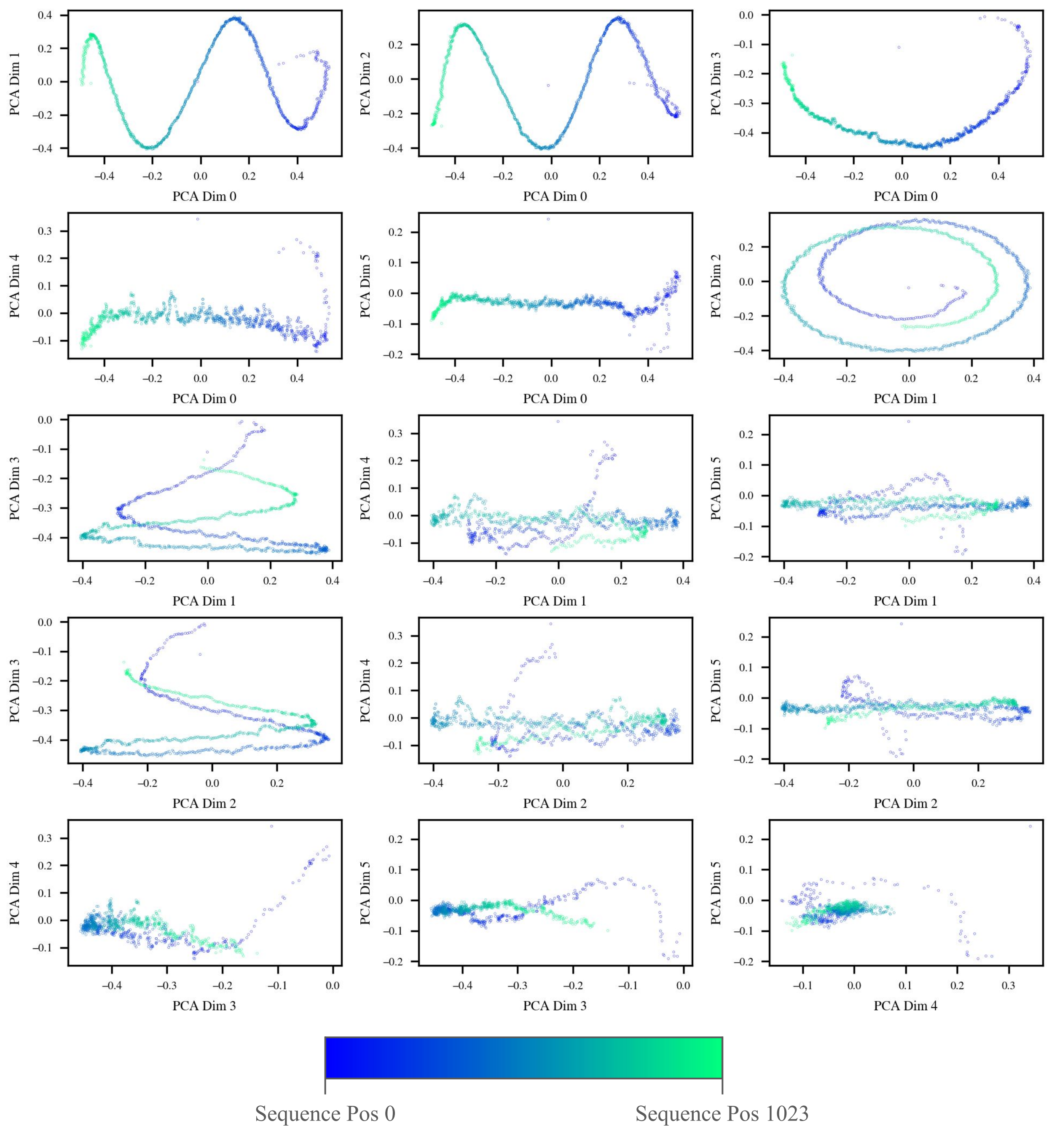

Figure 11: Visualization of post-add latent states from GPT-2 intermediate blocks after conversion to unit vectors, averaged across samples and layers. 15 plots visualize all unique pairs of 6 PCA dimensions.

As has already been established in previous works, the positional embeddings of GPT-2 form a sort of "helix" in high-dimensional space [19, 20]. This can clearly be seen in Figure 11. We also notice that a clear pattern across the sequence positions remains prominent for all combinations of PCA dimensions, demonstrating the high dimensionality of the geometric pattern formed by the GPT-2 positional embeddings.

4.5.2 LLaMa Positional Encodings

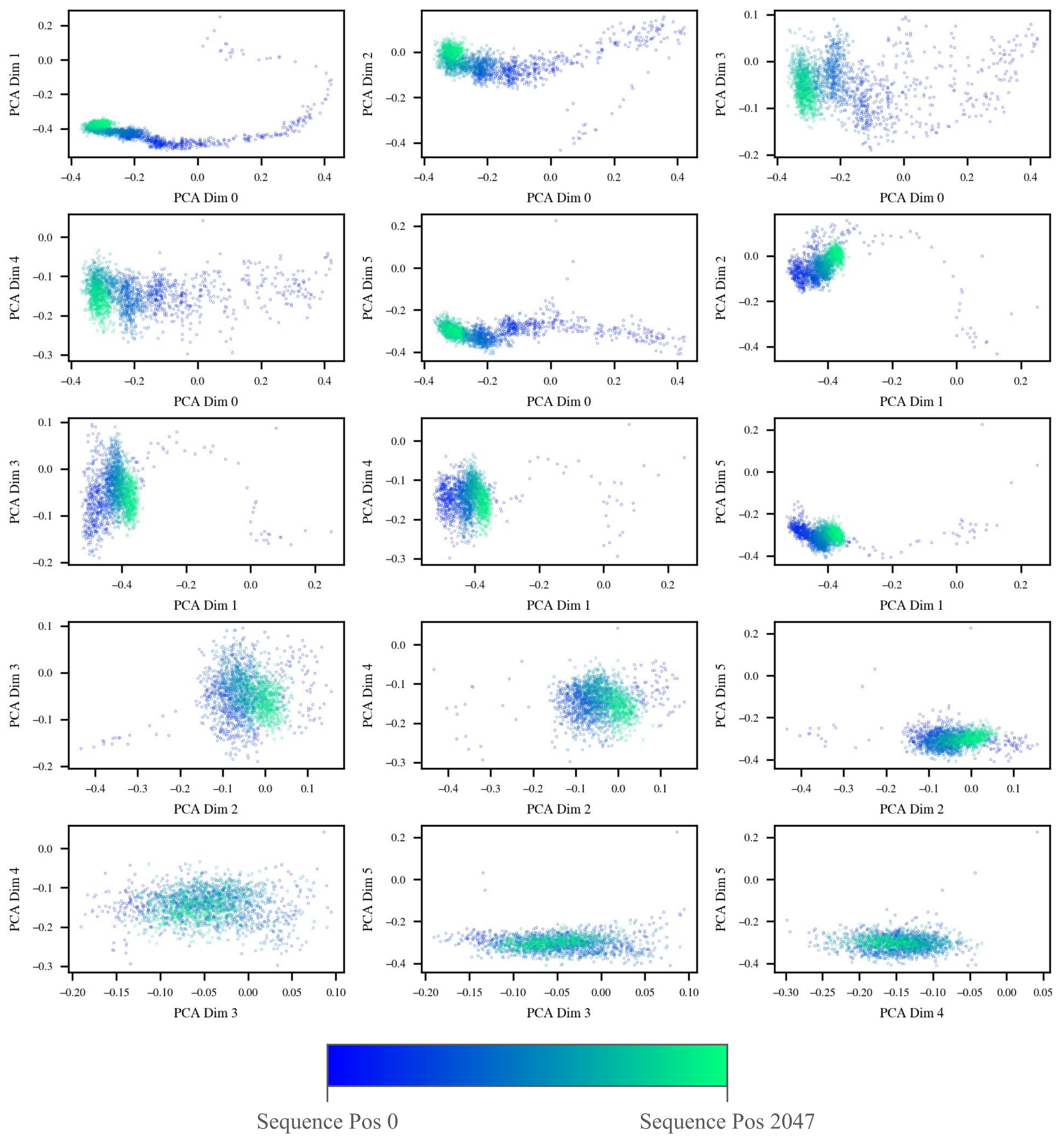

Figure 12: Visualization of post-add latent states from LLaMa intermediate blocks after conversion to unit vectors, averaged across samples and layers. 15 plots visualize all unique pairs of 6 PCA dimensions.

In Figure 12, we see a clear geometric pattern formed between sequence positions in subplots visualizing pairs of earlier PCA dimensions. In the first 3 rows of subplots, there appears to be a long "tail" of initial sequence positions which then leads into a dense cloud of latent states at later sequence positions. In subplots visualizing pairs of later PCA dimensions (the final 2 rows), this pattern disappears, and the latent states instead form a relatively uniform “cloud” with no significant pattern formed by the sequence positions. This may be an indication that the geometric pattern formed by the sequence positions is relatively low-dimensional, as it quickly disappears within the first few PCA dimensions.

Due to the inherent nature of RoPE applying only *within* self-attention heads, it is not straightforward to fully separate the effects of the RoPE-augmented attention heads on latent states from other factors such as the content of the input tokens themselves. This is unlike how it is possible to separate the learned positional embeddings of GPT-2 from any other context by simply analyzing the learned positional embeddings themselves. As such, it is not possible to determine via these visualizations alone whether the geometric patterns observed in Figure 12 are a result of RoPE, the relative convergence of features within contributions from self-attention heads as the number of previous tokens increases, or both. We leave interpreting the sequence-wise latent state geometric patterns of RoPE models to future research.

4.6 Repeating Token Experiment

We hypothesize that, in the case of RoPE-based models, the latent states of the same token repeated many times should eventually "converge" in latent space; after all, the attention heads of the model should view 2000 previous tokens as little different from how it views 2001 previous tokens. We experiment with inputs to LLaMa which are simply a sequence of the same token repeated many times. We only run this experiment for LLaMa as the same logic does not necessarily hold in the case of the learned positional embeddings of GPT-2, which introduces an absolute signature for each sequence position, whereas RoPE only affects *relative* distinctions between sequence positions.

We choose to repeat the token corresponding to the character "e" for the following reasons:

- It does not possess any particularly special syntactic function, unlike tokens such as "." or "\n" which often play a significant role in separating or demarcating sections of text with different contexts.
- "e" is not commonly repeated in long sequences. At the same time, it *is* occasionally repeated in text, such as in "weeeeeeeeeeeeeeeee". Thus, "e" serves as a good "neutral" token in the sense that a sequence of repeating "e" tokens is neither very common nor extremely perplexing.
- "e" by itself does not carry as much semantic information as other tokens which are complete words or concepts might. Of course, "e" can represent a mathematical constant; however, usually this would only become apparent when "e" is paired with other tokens of mathematical context.

We note that we tokenize the sequence of repeating "e" tokens with the `<BOS>` token as the initial token.

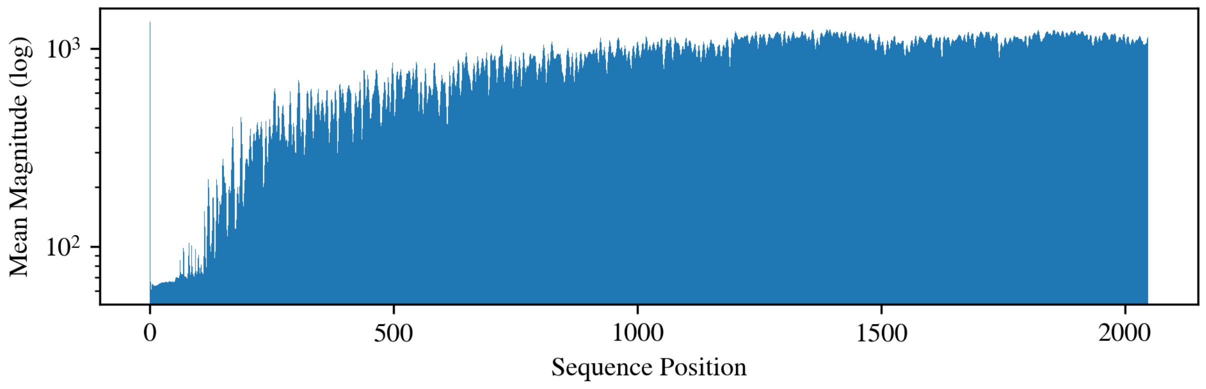

Figure 13: Latent state norm for each sequence position. The latent state norms were averaged across the intermediate layers.

Figure 13 shows the average latent state norm across the intermediate layers for each sequence position. After the initial spike of the initial `<BOS>` token, the norm of the repeating token latent states stays relatively low for ~ 100 tokens from gradually climbing to a large norm similar to that of the initial token latent states.

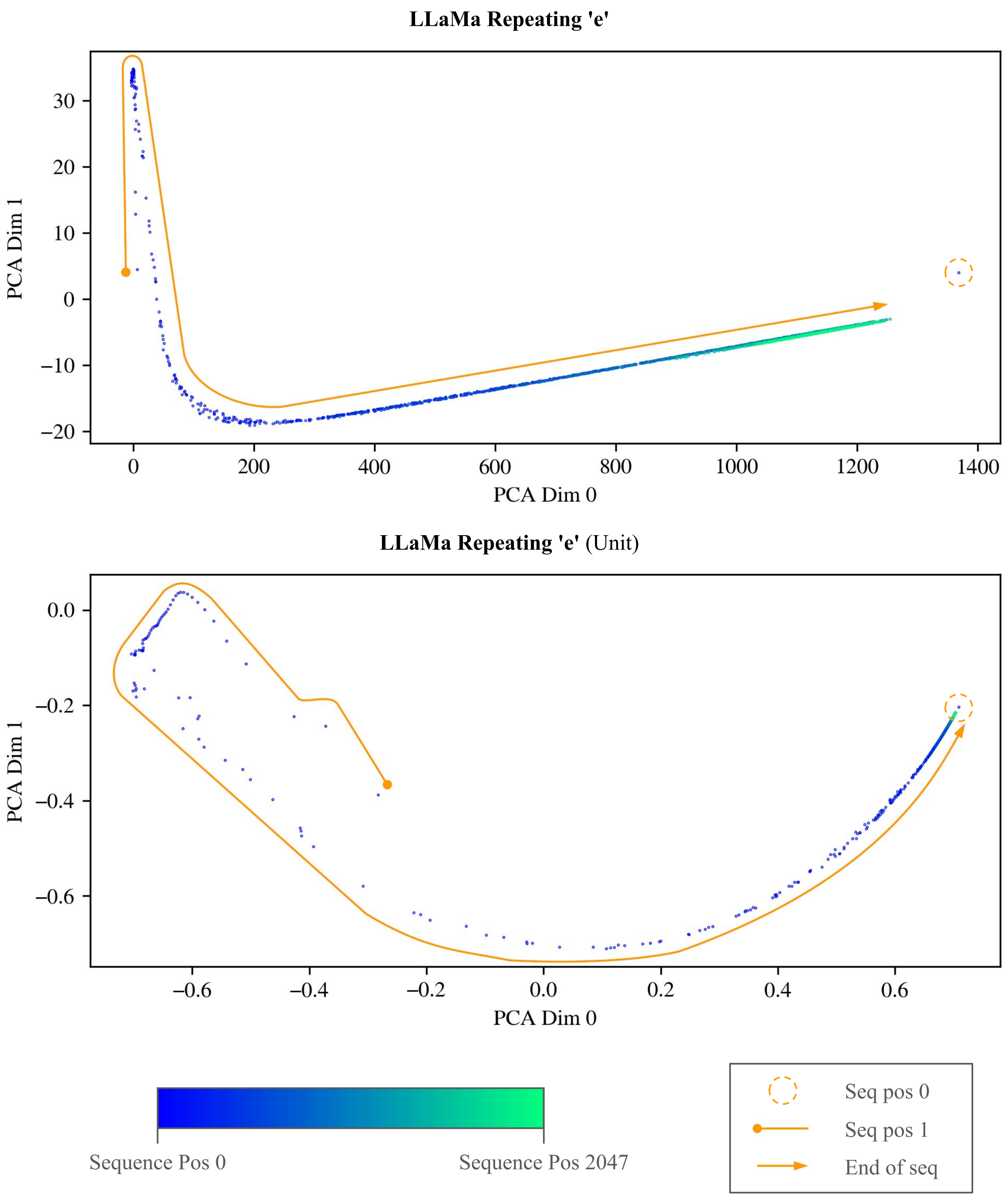

Figure 14: PCA of the repeating token latent states showing a convergence pattern. The latent states used were from the intermediate layers, and the latent states were averaged across the layers before dimensionality reduction or visualization.

Figure 14 shows a visualization after PCA of the repeating token latent states. As expected, a clear convergence pattern can be seen from the earlier to the later tokens. Notably, the initial (0-th) token is much closer to the final tokens than to the 1-st token. This is even true in the unit vector case, where only direction matters, and indicates that the later repeating "e" token latent states with high norm point in a similar direction to the initial `<BOS>` token. This is consistent with the findings of [17], which found that "massive activations" (what we refer to as the latent states with huge norms) tend to be constant (all point in a similar direction).

5 Conclusion

We present a study into visualizing the internal representations of Transformer-based language models. Through systematic analysis of GPT-2 and LLaMa, we demonstrate how dimensionality reduction techniques can reveal geometric patterns that show how these models organize and process information. Our experiments highlight several notable phenomena. Most significantly, we identify a persistent geometric separation between attention and MLP component outputs (section 4.4), a pattern that appears consistent across different model architectures and has not been previously documented to our knowledge. We also noted the high norm of latent states at the initial sequence position (section 4.2), a phenomenon that extends beyond special tokens like <BOS> to many vocabulary tokens in LLaMa despite its use of relative position encodings. Additionally, we visualized the layerwise evolution of latent states (section 4.3), the geometric effects of sequence position (section 4.5), and experimented with repeated token sequences (section 4.6). Ultimately, we add to the growing body of interpretability research by motivating further work into analyzing feature geometry. Future work can help further our understanding of the dynamics of latent states within Transformer models across layers, models, and experimental conditions. By deepening our understanding of how these geometric structures emerge and behave, we move closer to principled, reliable interpretability methods capable of guiding the development of more transparent and understandable models.

Acknowledgments

We thank the University of Virginia Research Computing and the Department of Computer Science for providing the computational resources that made this research possible.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[2] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[4] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[5] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[6] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models, 2024.

[7] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

[8] Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear, 2025.

[9] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[10] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[12] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019.

[13] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence, 2020.

[14] Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023.

[15] Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters, 2025.

[16] Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference?, 2025.

[17] Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models, 2024.

[18] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[19] Adam Scherlis. An exploration of gpt-2's embedding weights. `https://www.lesswrong.com/posts/BMghmAxYxeSdAteDc/an-exploration-of-gpt-2-s-embedding-weights`, Dec 2022.

[20] Adam Yedidia. Gpt-2's positional embedding matrix is a helix. `https://www.lesswrong.com/posts/qvWP3aBDBaqXvPNhS/gpt-2-s-positional-embedding-matrix-is-a-helix`, Jul 2023.

Glossary

0-Based Indexing — A convention where counting begins at zero, used for numbering Transformer blocks and sequence positions (e.g., "block 0" or "sequence position 0"). *Section 2.1*

Attention Component — A Transformer component that computes relationships and transmits information between tokens via self-attention, enabling context-dependent representations. *Section 2.1*

Feature Geometry — The structure and organization of learned representations within high-dimensional latent space, often analyzed through dimensionality reduction. *Section 1*

Initial Token — The first token in a sequence (position 0). *Section 4.2*

Intermediate Layers/Blocks — The middle portion of a Transformer, as defined in section 4.1. *Section 4.1*

Layer Dimension — The dimension of our generated dataset indexing Transformer's latent captures along the model's depth. *Section 3.1*

Layers / Components — The key submodules within each Transformer block: *normalization layers*, *attention*, and *MLP*. *Section 2.1*

Latent Space — The high-dimensional vector space in which latent states reside. *Section 2.1*

Learned Positional Encoding — A method of generating positional embeddings by learning weights. *Section 2.4*

Linear Representation Hypothesis (LRH) — The hypothesis that high-level features in language models correspond to approximately linear directions in representation space. *Section 2.2*

MLP (Multilayer Perceptron) — The feed-forward component of a Transformer block. *Section 2.1*

Norm — The L2 norm (Euclidean length/magnitude) of a latent vector. *Section 4.2*

Normalization Layer — A component (e.g., LayerNorm, RMSNorm) that stabilizes activations by rescaling and centering inputs; may appear before or after components in pre-norm or post-norm designs. *Section 2.1*

PCA (Principal Component Analysis) — A linear dimensionality reduction technique. *Section 2.3*

Positional Embedding — A positionally aware representation of token embeddings that encode absolute token order within the sequence. *Section 2.4*

Post-Add Latent States — Latent states captured after an attention or MLP output has been added back into the residual stream, reflecting updated representations. *Section 2.1*

Pre-Add Latent States — Output of an attention or MLP component before it is added back into the residual stream. *Section 2.1*

Residual Stream — The central communication channel formed by skip connections in the Transformer. *Section 2.1*

RoPE (Rotary Positional Encoding) — A method for generating *positional embeddings* by injecting position information through rotating attention query and key vectors in the attention mechanism according to token index. *Section 2.4*

Samples — Individual text inputs or tokenized passages passed through the model as batches to produce latent states for analysis. *Section 3.1*

Sample Dimension — The dimension of our generated latent state dataset indexing latent states generated from different input samples. *Section 3.1*

Sequence Dimension — The dimension of our generated latent state dataset indexing token positions within a sequence. *Section 3.1*

Skip Connections — Pathways that add each block's input back to its output, enabling a shared representation space across layers. *Section 2.1*

Unembedding Layer — The final linear layer that projects the model's last hidden state back into the vocabulary space. *Section 2.1*

UMAP (Uniform Manifold Approximation and Projection) — A nonlinear dimensionality reduction technique. *Section 2.3*

6 Appendix

6.1 Additional Sequence Position Visualizations

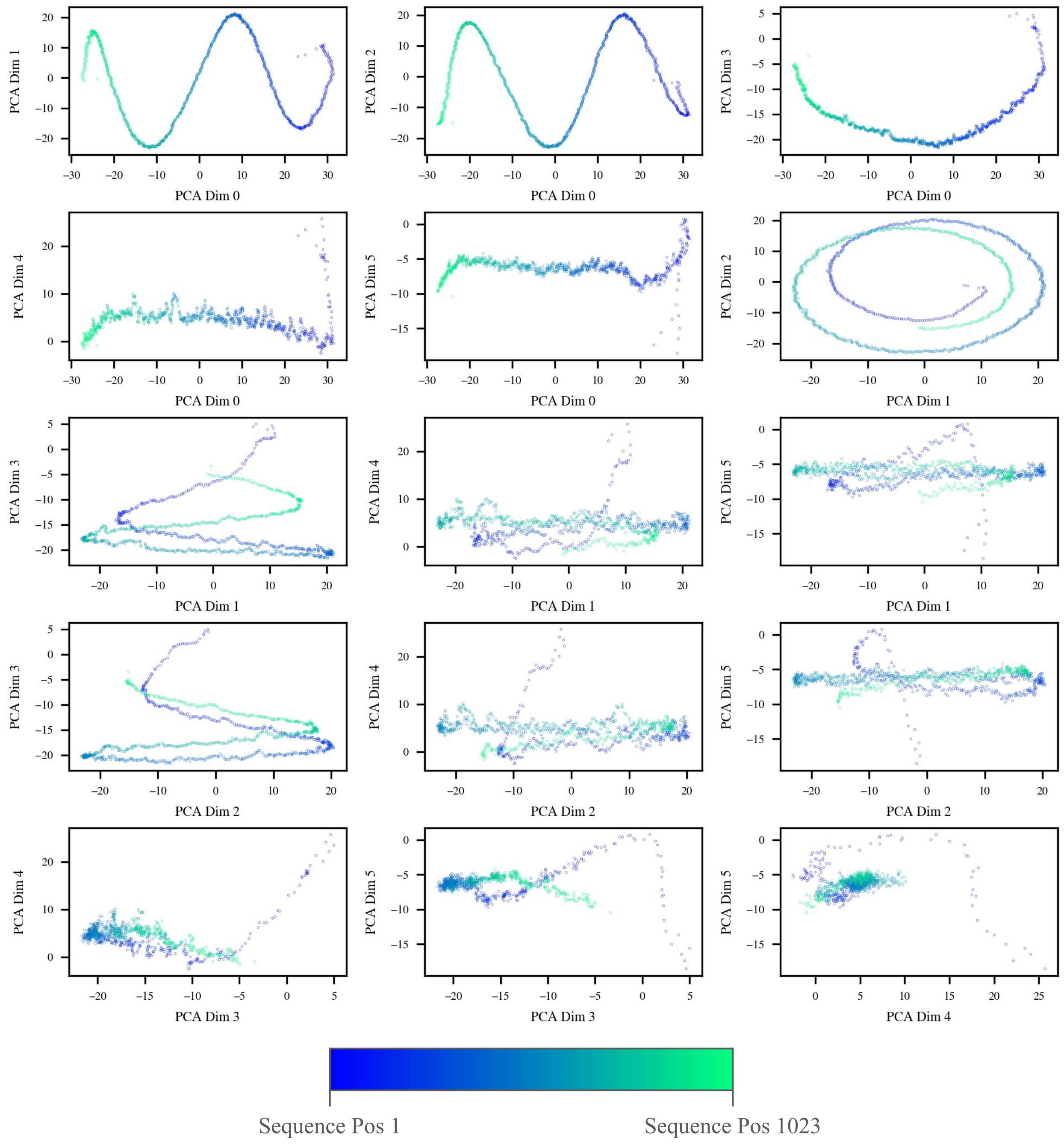

Figure 15: Visualization of post-add latent states from GPT-2 intermediate blocks, averaged across samples and layers. 15 plots visualize all unique pairs of 6 PCA dimensions.

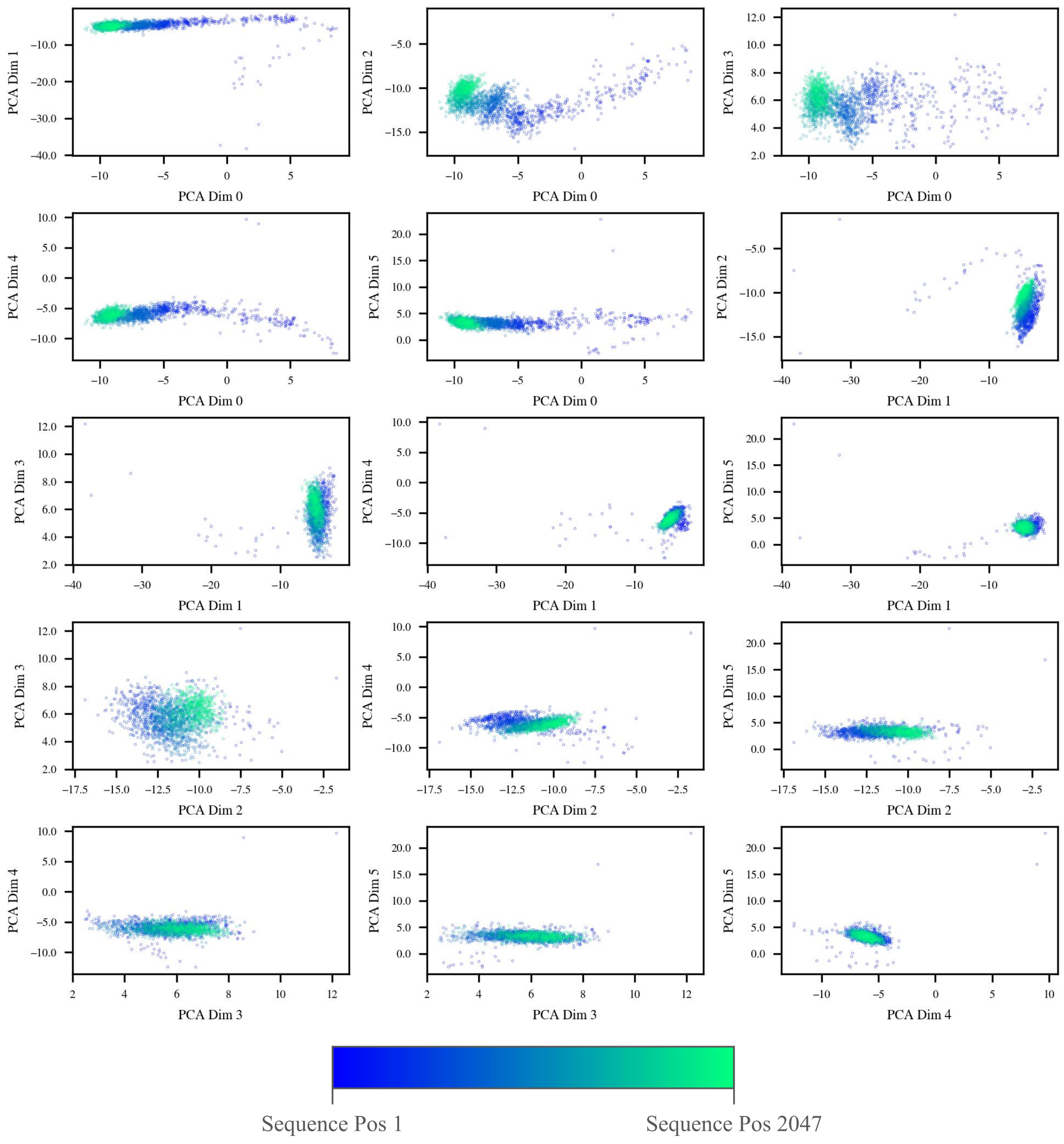

Figure 16: Visualization of post-add latent states from LLaMa intermediate blocks, averaged across samples and layers. 15 plots visualize all unique pairs of 6 PCA dimensions.