NERVE: NONLINEAR EIGENSPECTRUM DYNAMICS IN LLM FEED-FORWARD NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce NerVE, a unified eigenspectral framework for understanding how feed-forward networks (FFNs) in large language models (LLMs) organize and regulate information flow in high-dimensional latent space. Despite FFNs dominating the parameter budget, their high-dimensional dynamics remain poorly understood. NerVE addresses this gap through lightweight, memory-efficient tracking of eigenspectrum dynamics via four complementary metrics: Spectral Entropy (dispersion), Participation Ratio (effective dimensionality), Eigenvalue Early Enrichment (top-heaviness), and Jensen-Shannon divergence (distributional shifts). Our *key insight* is that FFN nonlinearities reinject and reshape variance across eigenmodes, fundamentally governing latent dimension utilization. We validate NerVE across model scales and diverse architectural configurations that each uniquely shape FFN dynamics: normalization strategies (PreLN, PostLN, MixLN, Norm-Free) controlling variance flow; FFN weight geometries constraining latent space; positional encoding and activation functions modulating information propagation. Across these settings, NerVE consistently recovers stable spectral signatures that correlate with model's generalization ability and respond predictably to design choices, providing actionable insights for architectural optimization beyond trial-and-error.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language tasks, driven in part by advances in transformer-based architectures. While much emphasis has been devoted to understanding attention mechanisms and token-wise interactions, the role of feed-forward networks (FFNs), particularly their nonlinear components, remains underexplored, despite FFNs dominating both the parameter budget and computational footprint of transformer-based models Geva et al. (2021); de Vries (2023).

Despite their apparent simplicity, FFNs perform high-dimensional *nonlinear* transformations that regulate information flow by reorganizing, compressing, and propagating the information extracted by attention modules across layers. Understanding how these transformations evolve and interact with architectural design choices remains a fundamental open question.

One challenge in interpreting FFNs is the absence of systematic and efficient tools for characterizing how latent representations are structured and transformed by nonlinear activations. Unlike self-attention, whose weight matrices make token-token interactions relatively easy to probe, FFN transformations unfold in a high-dimensional feature space that is far less accessible for direct visualization and probing. Prior work Kobayashi et al. (2024) used attention maps to study the input-contextualization effect of FFNs; however, this lens does not reveals how nonlinearity redistributes variance in the latent space, and misses the rich geometric structure inherent in these transformations.

To this end, we introduce NerVE, a unified, online, and memory-efficient framework for analyzing FFN latent geometry through the eigenspectrum analysis. NerVE summarizes pre- and post-activation spectra using four scale-invariant, distribution-aware metrics: spectral entropy (dispersion vs uniformity), participation ratio (effective latent dimensionality), eigenvalue early enrichment (top-heaviness), and Jensen-Shannon divergence (distributional shift).

From a methodological standpoint, these metrics span a broad theoretical range and expose the complementary facets of the eigenspectrum that any single scalar would obscure, thereby enabling

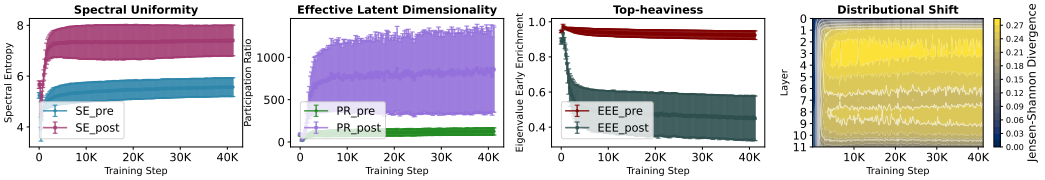

Figure 1: Nonlinear eigenspectrum dynamics in FFNs (GPT-2): FFN nonlinearity (GELU) regulate information flow by reinjecting variance, reactivating under-utilized directions (post-activation SE↑ and PR↑), and flattening the eigenspectrum, less top-heavy (post-activation EEE↓), in the high-dimensional latent space. The JS heatmap shows a depth-localized transition band where redistribution is strongest. NerVE quantifies these effects via scale-invariant: SE, PR, EEE, and JS.

continuous tracking of latent geometric dynamics throughout training. Spectral Entropy (SE) captures the uniformity of variance distribution (De Domenico & Biamonte, 2016; Garrido et al., 2023); whereas, Participation Ratio (PR) reflects the geometric notion of effective dimensionality, indicating how many directions meaningfully contribute to total variance Gao et al. (2017). Unlike SE and PR, Eigenvalue Early Enrichment, which quantify the top-heaviness, can distinguish the eigenspectrum t utilizing different fractions of the high-dimensional latent space (Marbut et al., 2023). Finally, the Jensen-Shannon (JS) divergence provides an information-theoretic distance measure between two eigenspectra, quantifying how nonlinear transformations redistribute variance (Lin, 1991).

We apply this framework across a diverse range of architectural settings, including LayerNorm placements: PreLN, PostLN, and MixLN (Li et al., 2025); normalization-free variants (Jha & Reagen, 2024; He & Hofmann, 2024); FFN weight-geometry constraints, including weight normalization Salimans & Kingma (2016), spectral normalization (Miyato et al., 2018), and hyperspherical constraints Liu et al. (2017); and positional encoding scheme (RoPE) Su et al. (2024).

Across settings, *a clear pattern emerges:* FFN nonlinearities do not merely rescale the activations, they actively reinject the variance into high-dimensional latent space and reawakens the inactive directions, and flatten the eigenspectrum by reducing their top-heaviness. As shown in Figure 1, the post-activation spectra in GPT-2 (125M) consistently show increases in SE and PR, and decreases in EEE, while JS heatmaps reveal depth-localized transition bands where redistribution is strongest. These findings highlight the active role of FFN nonlinearities in regulating information flow and latent dimensionality that downstream layers exploit.

Contributions: Our contributions can be summarized as follows:

1. **Framework.** We propose NerVE, a lightweight and memory-efficient methodology for online tracking of FFN eigenspectrum dynamics, using four distribution-aware, scale-invariant metrics.
2. **Conceptual insights.** We demonstrate that FFN nonlinearities do not simply rescale the activations but actively reorganize eigenspectra, reinjecting variance into under-utilized directions, flattening top-heavy distribution, thus regulating latent dimensionality available to later layers.
3. **Architectural causality.** We show that the architectural design choices—normalization layer placement, activation functions, gating, weight geometry, positional encodings—imprint early-emerging spectral signatures that can serve as reliable proxy for model's generalization ability.
4. **Empirical.** We validate NerVE on GPT-2 and LLaMA models trained from scratch on CodeParrot HuggingFace, OpenWebText, and C4 datasets Raffel et al. (2020), with FFN width sweeps.

2 NerVE: A Principled Framework for Eigenspectrum Analysis

Notations. Let L be the number of layers, d the embedding dimension, D the FFN hidden dimension, B the batch size, S the context length, and Σ the FFN (pre/post-activation) covariance matrix.

2.1 Formulation of Eigenspectrum-Based Framework

To understand how information is structured and propagated through LLM latent space, we analyze: (1) variance distribution across the eigenspectrum and its impact on effective dimensionality; (2) how nonlinearity within layer reshape this distribution; (3) how these patterns evolve across network depth and training. Our Geometric framework consists of four main components: i) activation collection, (ii) covariance matrix computation, (iii) eigendecomposition, and (iv) spectral metrics calculation.

Activation collection. For a FFN with (non-gating) architecture $\text{FFN}(x) = W_{\text{down}}\sigma(W_{\text{up}}x + b_1) + b_2$, where σ is the activation function (*e.g.*, ReLU, GELU), we collect $\text{PreAct}(X) = W_{\text{up}}x + b_1$ and $\text{PostAct}(X) = \sigma(W_{\text{up}}x + b_1)$, the output of the up projection, and input to the down projection (after activation function), respectively. For activation with gating mechanisms (e.g., SwiGLU in LLaMA), the architecture becomes $\text{FFN}(x) = W_{\text{down}}(\sigma(W_{\text{gate}}x) \odot (W_{\text{up}}x))$, where $\odot$ represents element-wise multiplication, and we collect $\text{PreAct}(X) = W_{\text{gate}}x$ and $\text{PostAct}(X) = \sigma((W_{\text{gate}}x) \odot (W_{\text{up}}x))$.

Covariance matrix computation. At the logging step t, for each layer l, we collect full activation matrices $\text{PreAct}(X^{(l,t)}) \in \mathbb{R}^{N\times D}$ and $\text{PostAct}(X^{(l,t)}) \in \mathbb{R}^{N\times D}$, where $N = B \times S$ is the total number of tokens in the batch. These tensors, originally shaped $[B, S, D]$, are flattened to $[B \times S, D]$, intentionally discarding sequence order. This allows us to compute an unbiased covariance matrix for all tokens in the batch, treating each token as an independent sample in embedding space.

Computing the covariance using all N tokens without any sub-sampling, ensures *exact* second-order statistics of the batch rather than their statistical approximations, and spectral analysis captures the *true* statistical properties of the activation distributions. For each set of activations, we compute an unbiased sample covariance matrix as follows:

$$\Sigma = \frac{(X-\mu)^T(X-\mu)}{N-1} \quad \in \mathbb{R}^{D\times D,} \text{ where } X \in \mathbb{R}^{N\times d} \text{ are activations and } \mu = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (1)$$

This yields two covariance matrices per FFN layer: $\Sigma^{(l,t)}_{\text{PreAct}}(X)$ and $\Sigma^{(l,t)}_{\text{PostAct}}(X)$.

Eigendecomposition For each covariance matrix, we perform eigendecomposition to obtain the eigenvalues $\Sigma v = \lambda v$. We use `torch.linalg.eigvalsh` for numerical stability, as covariance matrices are symmetric positive semi-definite. The resulting eigenvalues are sorted in descending order: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D \geq 0$. We define $\Lambda = \sum_{i=1}^{D} \lambda_i$, the total variance, and normalized eigenvalues to create a probability distribution as $\tilde{\lambda}_i = \lambda_i/\Lambda$.

Spectral metrics computation Next, we compute four scalar metrics from the eigenspectrum of $\Sigma^{(l,t)}_{\text{PreAct}}(X)$ and $\Sigma^{(l,t)}_{\text{PostAct}}(X)$, which quantify distinct aspects of the eigensectrum dynamics: Spectral Entropy, Participation Ratio, Eigenvalue Early Enrichment, and the Jensen-Shannon divergence.

2.2 Eigenspectrum Metrics for Analyzing High-Dimensional Latent Space

Spectral Entropy (SE) Spectral Entropy quantifies the uniformity of eigenvalue distribution in high-dimensional latent spaces. Formally, it is the Shannon entropy of the normalized eigenvalue distribution derived from a layer's covariance matrix: $\text{SE} = -\sum_{i=1}^{D} \tilde{\lambda}_i \log \tilde{\lambda}_i$.

Mathematically, spectral entropy is equivalent to the von Neumann entropy (vNE) in quantum information theory, which quantifies the degree of quantum entanglement or *mixedness* of a quantum state (De Domenico & Biamonte, 2016). In quantum mechanics, vNE is defined as: $S_{\text{vNE}}(\rho) = -\text{Tr}(\rho \ln \rho)$, where ρ denotes a density matrix, a positive semidefinite operator with unit trace that encapsulates the probabilistic nature of quantum states (Nikitin et al., 2024).

For FFN, an analogous density matrix is created by normalizing the covariance matrix by its trace: $\rho_{\text{FFN}} = \frac{\mathbf{\Sigma}}{\text{Tr}(\mathbf{\Sigma})}$, where $\text{Tr}(\mathbf{\Sigma}) = \Lambda$. Applying this to ρ_{FFN}, SE becomes the Shannon (or von Neumann) entropy of the normalized eigenvalue distribution $\text{SE} = -\sum_{i=1}^{D} \tilde{\lambda}_i \log \tilde{\lambda}_i$.

Thus, when the eigenspectrum exhibits significant anisotropy (e.g., $\lambda_1 \gg \lambda_2, \ldots, \lambda_D$), SE approaches zero, indicating a collapsed or low-rank representation. Conversely, when eigenspectrum approach uniformity ($\lambda_i \approx \lambda_j : \forall i, j$), SE approaches its theoretical maximum, $\ln(D)$.

Participation Ratio (PR). It measures *effective dimensionality* of an eigenspectrum (Hu & Sompolinsky, 2022) and quantifies how many dimensions significantly hold variance. Formally,

$$\text{PR} = \frac{\left(\sum_{i=1}^{D} \lambda_i\right)^2}{\sum_{i=1}^{D} \lambda_i^2} = \frac{\Lambda^2}{\sum_i \lambda_i^2}, \quad \text{where} \quad 1 \leq \text{PR} \leq D \quad (2)$$

PR values close to 1 indicates maximal anisotropy (i.e., variance concentrated in a single direction), while a value near D indicates uniform variance across all dimensions. While SE depends on the

entire distribution shape (including small eigenvalues) and measures the uniformity of distribution, PR focuses on how many directions are meaningfully active.

Notably, different eigenvalue distributions can share the same SE and PR, yet allocate variance very differently across the eigenspectrum. hence, we need to include additional metric that reliably differentiate spectra that utilizing the latent space in qualitatively different ways.

Early Eigenvalue Enrichment (EEE). It quantifies the *top-heaviness* of an eigenspectrum by tracking how rapidly the leading principal directions accumulate variance. Specifically, it captures how *front-loaded* the variance is among the top eigenvalues, by assessing how quickly the cumulative sum surpasses that of a uniform spectrum (Marbut et al., 2023).

Formally, the proportion of variance explained by the top k principal directions is defined by the normalized cumulative sum at index k as $\widetilde{S}_k = \frac{1}{\Lambda}\sum_{i=1}^{k}\lambda_i$, and for comparison, the ideal uniform reference grows linearly as $\frac{k}{D}$ (see Figure 2). The EEE score is then the average vertical distance between the empirical cumulative curve and this ideal line, normalized by the maximal possible value (which occurs when all variance is in a single component):

$$\mathrm{EEE} = \frac{1}{\frac{1}{2}D}\sum_{k=1}^{D}\left(\widetilde{S}_k - \frac{k}{D}\right) = 2 \times \sum_{k=1}^{D}\left(\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{D}\lambda_i} - \frac{k}{D}\right) \times \frac{1}{D}. \tag{3}$$

$\mathrm{EEE} \approx 1$ indicates that most of the variance is concentrated in the top few directions, forming a steep eigenvalue spectrum; conversely, $\mathrm{EEE} \approx 0$ corresponds to a nearly uniform spectrum, where variance accumulates gradually across all dimensions.

We analyze the cumulative variance distribution of various eigenspectra over a 768-dimensional latent space, using the EEE metric in Figure 2. The resulting curves shows a spectrum of dimensional utilization, ranging from extreme anisotropy to fully uniform variance. The One dimension spectrum exhibits an EEE of 1.00, where nearly all variance is concentrated in a single dominant principal component, indicative of a highly *degenerate* latent representation. As more dimensions begin to carry variance (from 10% to 99%), the EEE value decreases from 0.94 to 0.44, suggesting a gradual transition toward more distributed representations. Notably, EEE's nonlinear scaling with dimension count highlights its sensitivity to early eigenvalue dominance, making it a valuable diagnostic for understanding how architectural choices and training dynamics shape the effective dimensionality of latent spaces.

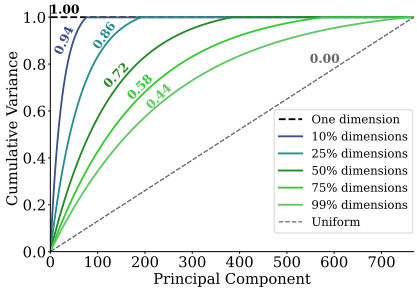

Figure 2: Cumulative variance distribution across a 768-dimensional latent space. Higher values (shown on curves) indicates top-heavy concentration in a few dominant directions, while lower one reflects a more uniform distribution.

Jensen-Shannon Divergence (JS) Unlike the previous metrics, which describe a single eigenspectra in isolation, JS provides a principled measure of dissimilarity between two eigenspectrum within a layer. Specifically, it quantify the *extent of distributional shifts* from the *pre* to *post* eigenspectrum caused by FFN nonlinearity. For a normalized eigenvalue distributions $P_{\text{pre}} = \{\hat{\lambda}_i^{\text{pre}}\}_{i=1}^{D}$ and $P_{\text{post}} = \{\hat{\lambda}_i^{\text{post}}\}_{i=1}^{D}$, where $\hat{\lambda}_i = \lambda_i / \sum_{j=1}^{D}\lambda_j$, the JS is defined as (see Amari (2016, Chapter 4.6.3)):

$$\mathrm{JS}(P_{\text{pre}} \parallel P_{\text{post}}) = \frac{1}{2}D_{\mathrm{KL}}(P_{\text{pre}} \parallel M) + \frac{1}{2}D_{\mathrm{KL}}(P_{\text{post}} \parallel M) \tag{4}$$

where $M = \frac{P_{\text{pre}}+P_{\text{post}}}{2}$ is the midpoint distribution and D_{KL} is Kullback-Leibler divergence:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{i=1}^{D}\hat{\lambda}_i^{P}\log\left(\frac{\hat{\lambda}_i^{P}}{\hat{\lambda}_i^{Q}}\right) \tag{5}$$

For numerical stability, we compute JS for FFN as follows:

$$\mathrm{JS}(P_{\text{pre}} \parallel P_{\text{post}}) = \frac{1}{2}\sum_{i=1}^{D}\hat{\lambda}_i^{\text{pre}}\log\left(\frac{2\hat{\lambda}_i^{\text{pre}}}{\hat{\lambda}_i^{\text{pre}}+\hat{\lambda}_i^{\text{post}}}\right) + \frac{1}{2}\sum_{i=1}^{D}\hat{\lambda}_i^{\text{post}}\log\left(\frac{2\hat{\lambda}_i^{\text{post}}}{\hat{\lambda}_i^{\text{pre}}+\hat{\lambda}_i^{\text{post}}}\right) \tag{6}$$

The key benefits of above formulation of JS divergence, compared to a simpler KL divergence, for eigenspectrum analysis are: (1) *Symmetry*: $\text{JS}(P_{\text{pre}} \parallel P_{\text{post}}) = \text{JS}(P_{\text{post}} \parallel P_{\text{pre}})$ enables unbiased comparison of pre-activation and post-activation eigenspectrum, whereas KL is asymmetric and would prioritize one distribution as the reference; (2) *Boundedness and interpretability*: $0 \leq \text{JS}(P_{\text{pre}} \parallel P_{\text{post}}) \leq ln(2)$, facilitating standardized comparisons across layers of different dimensions, unlike KL which could become unbounded. Moreover, the bounded scale makes JS values interpretable under distributional shift, unlike KL which could yields unbounded values, in isolation, and (3) *Numerical stability*: JS offers superior numerical stability when analyzing eigenspectra with near-zero eigenvalues, which are common in neural network representations.

In the context of FFNs, JS divergence quantifies the information-theoretic distance between pre-activation and post-activation eigenspectra, identifying FFNs where nonlinearity causes significant geometric restructuring. Large JS values indicate that nonlinear activations substantially redistribute variance across different principal components, potentially creating new directions of specialization or eliminating others. Conversely, small JS values suggest that nonlinearities primarily rescale existing directions without fundamentally altering the latent space geometry. Refer to Appendix B, for a discussion on the distributional sensitivity of these spectral metrics.

3 Experimental Results

Models and datasets We evaluate the FFN eigenspectrum of two model families: GPT-2 and LLaMA-style architectures. For GPT-2, we train a 125M parameter model on 2.1B tokens from the CodeParrot dataset, which is created from 20M GitHub Python files and preprocessed using HuggingFace tokenizer of vocabulary size 50K. For LLaMA-style models, we train in-house variants with 71M and 130M parameters on the C4 dataset, tokenized using the T5-base tokenizer with a 32K vocabulary. These LLaMA variants follow the architectural specifications (depth, embedding dimensions, FFN width, positional encoding, and SwiGLU activation) from Li et al. (2025), which adopts downscaling methodology of Lialin et al. (2024); Zhao et al. (2024). For experiments with RoPE, we train GPT-2 on OpenWebText dataset, following the architectural settings and training recipe from Loshchilov et al. (2025).

Training setup All experiments are conducted on NVIDIA RTX 3090 GPUs (24 GB). GPT-2 models are trained for 41K steps with context length 128 on the CodeParrot dataset. For RoPE experiments, GPT-2 is trained on 26B tokens from OpenWebText using 4 GPUs with context length 512. LLaMA-71M is trained on 1.1B tokens for 10K steps, while LLaMA-130M variants are trained on 2.2B tokens for 20K steps. All LLaMA models use a context length of 256.

3.1 FFN Nonlinearity Reinject Variance and Flatten the Eigenspectrum

Variance is reinjected, not merely rescaled Figure 1 contrasts pre- and post-activation spectral dynamics and highlights the role of nonlinearity withing FFN. The PreAct eigenspectrum is highly top-heavy as most variance is concentrated in a few leading directions. This is reflected by lower SE and PR, indicating a lower utilization of the latent space. Once the nonlinearity is activated, both SE and PR jump upward across training, suggesting that the nonlinearity redistributes variance across more dimensions. In effect, the nonlinearity *reawakens* previously inactive directions, injecting new degrees of freedom into the latent space. This reinjection of variance promote the disentanglement of features which facilitate more effective downstream processing in subsequent layers, and enables deeper layers to operate on richer and more informative representations.

Flattening and reshaping the eigenspectrum The variance redistribution has a noticeable impact on the spectrum shape. The EEE values, which quantifies how sharply leading eigenvalues dominate, drops consistently for post-activation, re-affirming that the spectrum is being flattened. Instead of concentrating variance in a small number of dominant modes, the post-activation spectrum spreads variance more evenly. Moreover, the JS heatmaps shows a distributional shift: post-activation eigenspectra are not merely scaled versions of pre-activation ones but are effectively reordered.

GELU vs. ReLU: Similar Trajectory, Distinct Dynamics GELU (Figure 1) and ReLU (Figure 3) follow the same qualitative trajectory—variance reinjection (↑SE, ↑PR), spectral flattening (↓EEE), and distributional reordering (↑JS)—but differ in pace and extent. While ReLU stabilize SE and PR earlier, suggesting a faster reinjection of variance, GELU progresses more gradually yet ultimately

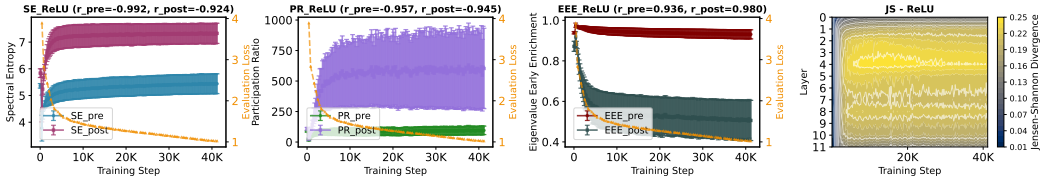

Figure 3: Eigenspectrum dynamics illustrate *how FFN nonlinearities regulate information flow and reshape the eigenspectrum* during training for GPT-2 (ReLU) on CodeParrot. Pre- and post-activation dynamics are shown for SE, PR, and EEE, highlighting how nonlinearities reinject variance and alter spectral structure. JS heatmaps (rightmost) capture the layer-wise distributional shift induced by nonlinearity. In-panel titles report Pearson correlations (r) between each metric and evaluation loss.

pushes PR_{post} to higher values, indicating that its smoother nonlinearity enables exploration of a broader subspace. This broader exploration correlates with GELU's lower perplexity, underscoring its effectiveness despite slower dynamics.

Spectral dynamics correlate with generalization We quantify how eigen-metrics shift in their favorable directions (SE ↑, PR ↑, EEE ↓) correlate with next-token prediction performance. We observe a strong and consistent correlation with validation loss ($|r| \geq 0.94$), suggesting that optimizer actively leverages the spectral shifts to improve generalization.

Together, these eigen-metrics highlights the functional role of nonlinearity: (1) improving the directional usage of FFN latent space (SE ↑, PR ↑), (2) flattening eigenspectrum (EEE ↓), and (3) inducing a shift in the geometry of latent representations (JS ↑). This serve as the geometric underpinning of the nonlinear expressivity principles in transformer models.

3.2 Compensatory Role of FFN Nonlinearity in the Absence of LayerNorms

Removing LayerNorm from transformer architectures eliminates their layerwise re-centering and variance normalization, shifting the burden of statistical regularization entirely onto the attention and FFN sub-blocks. This motivates a central question: *Can FFN activation functions compensate for the absence of normalization, and if so, to what extent and through what mechanisms?* Our findings reveal that, unlike GELU, ReLU-family activations actively compensate the absence for removal of LayerNorms by regulating the FFN latent space variance.

Spectral inertia in normalization-free GELU models Normalization-free GELU model exhibits spectral inertia in early layers, characterized by $\text{EEE}_{\text{post}} \approx 1$ and JS ≈ 0 (see Figure 4). This indicates that the nonlinearity in early FFNs fails to reinject variance into the latent space, leaving the eigenspectrum heavily front-loaded. Consequently, variance remains confined to a few dominant subspaces, and there is a significant overlap between SEpre and SEpost. Thus, nonlinearity in early FFNs does not activate new directions, and information continues to flow through a narrow subspace in subsequent layers. This *spectral bottleneck* reflects a downstream consequence of entropic overload—a critical failure mode observed in normalization-free LLMs Jha & Reagen (2024)—where a disproportionate number of attention heads in the early layers remains in persistently high-entropy states throughout training, squandering the representation diversity of multi-head attention mechanism. Ultimately, this degrades the performance and leads to a higher perplexity.

Early FFNs overcompensate to break spectral inertia in normalization-free ReLU models In contrast with GELU, ReLU and learnable-slope Leaky ReLU variant exhibit strong compensatory behavior when LayerNorms are removed. Specifically, in the first two FFN layers, the post-to-pre Participation Ratio (PR) gain surges by $\approx 20\times$ to $300\times$ (blue curves, Figure 4), indicating an abrupt reinjection of variance into previously underutilized latent directions. Consequently, the post-activation EEE_{post} remains consistently low ($\approx$ 0.3-0.5) across layers, indicating that the spectrum becomes flatter and more isotropic, rather than top-heavy. This redistribution is further corroborated by non-overlapping SEpre and SEpost spectrum, and by JS peaks $\approx$0.48 in the early-layer contour maps, confirming the crucial role of nonlinearity in reshaping eigenspectrum in early FFNs

This *aggressive variance injections* demonstrate that FFN nonlinearity can partially assume the statistical regularization role of LayerNorm, widening the latent manifold and mitigating spectral bottlenecks. In terms of predictive performance, both ReLU variants reduce the perplexity gap to the LayerNorm baseline by $\approx$50% (refer to Table 1).

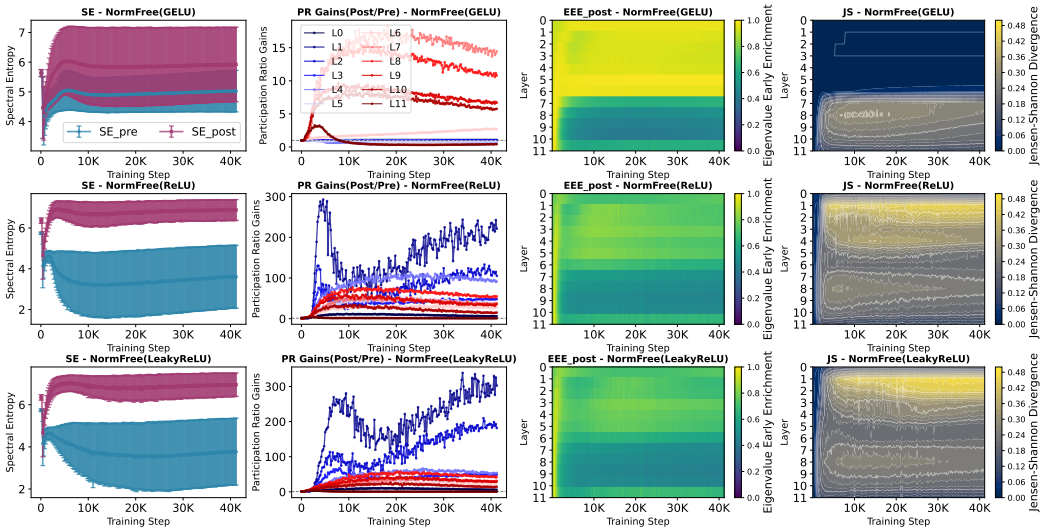

Figure 4: Eigenspectrum dynamics for norm-free GPT-2 models using GELU (top), ReLU (middle), and learnable-slope Leaky ReLU (bottom). Columns show layer-averaged SE (pre vs. post), PR gain (post to pre), post-activation EEE (yellow $\rightarrow$ top-heavy), and JS (yellow $\rightarrow$ strong redistribution) across layers and training steps. Norm-free GELU exhibits spectral inertia in layers 0 to 5 (EEE $\rightarrow$ 1, JS $\rightarrow$ 0); whereas, *ReLU and Leaky ReLU aggressively reinject variance* (PR gain $>$ **200**$\times$) and flattening the spectrum (EEE < 0.3).

3.3 Feed-Forward Networks Weight Geometry and Eigenspectrum Dynamics

Previously, we have seen that how ReLU variants improve the redistribution of top-heavy eigenvalues in the early layers of normalization-free LLMs. We now analyze how parametric normalization applied to their FFNs further influence eigenspectrum dynamics. Figure 5 shows the effects of weight, spectral, and hyperspherical normalization applied to FFNs.

Table 1: Evaluation perplexity (PPL $\downarrow$) comparison across GPT-2 baseline models (GELU and ReLU), norm-free models (GELU, ReLU, learnable-slope Leaky ReLU). Parametric normalization (Weight, Spectral, Hyperspherical) are applied to FFNs of norm-free learnable-slope Leaky ReLU models. All models trained on 2.1B tokens from CodeParrot dataset.

	Baseline Models		Norm-free Models			Norm-free w/ FFN-Norm		
	GELU	ReLU	GELU	ReLU	Leaky ReLU	WNorm	SNorm	HNorm
PPL	2.714	2.774	3.223	2.988	3.081	3.041	3.000	3.122

Parametric normalization alters the localization of distributional shifts across layers Despite being applied only to FFN linear layers, each parametric normalization technique induces distinct learning dynamics, as demonstrated by the layerwise JS divergence in Figure 5 (rightmost column). Specifically, SNorm exhibits highly localized distributional shifts in the mid-to-deeper layers that emerge very early in training. In contrast, WNorm induces distributional shifts in a smaller subset of mid layers that appear very late in training. Meanwhile, HNorm triggers strong shifts in the early layers at the very-beginning of training, which gradually diminish as training progresses.

Spectral normalization achieves superior performance through smooth and sustained spectral flattening By constraining the spectral norm of each FFN weight matrix, SNorm induces early and consistent spectral flattening, reflected in uniformly negative ΔEEE (Post-Pre) values in Figure 5, especially in deeper layers. This yields the lowest EEE_post ($\approx$-0.45) among all parametric normalization methods, indicating balanced variance distribution across a moderate number of directions (PR_post $\approx$ 200) and improved latent space utilization. In contrast, WNorm shows delayed and highly localized flattening in a few mid layers, while HNorm induces early flattening in shallow layers that vanishes as training progresses.

Hyperspherical normalization underperforms due to early overshooting in eigenspectrum HNorm projects weight vectors onto a unit hypersphere, which rapidly expands latent capacity, indicated by a sharp increase in PR_post (exceeding 600). However, this expansion leads to an

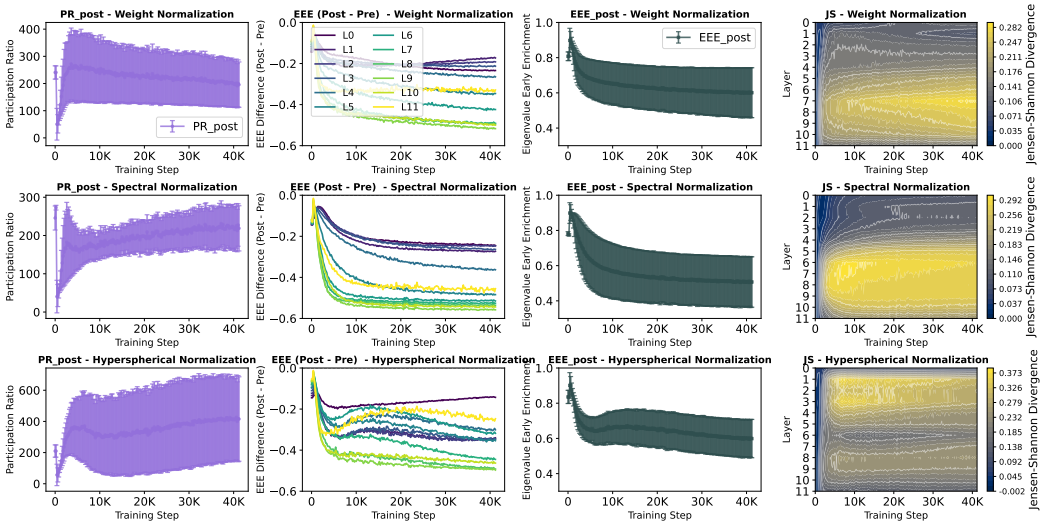

Figure 5: Impact of FFN (parametric) normalization in norm-free GPT-2 with learnable-slope leaky ReLU. Eigenspectrum dynamics are quantified by latent capacity (PR_post), spectral regularization and flattening (ΔEEE and EEE_post), and distributional shift (JS). Top to bottom: Weight, Spectral, and Hyperspherical Normalization. Each method exhibits distinct JS localization and spectral patterns, showing different influences on FFN internal dynamics.

early-overshooting in ΔEEE dynamics, and EEE_post values remain high across depth, indicating the persistence of dominant directions despite the expanded capacity. This behavior is also reflected in JS divergence patterns, suggesting an undifferentiated and inefficient use of model's depth. Hence, the combination of early overshooting, lack of spectral control, and depthwise redundancy likely, leads to HNorm's degraded perplexity. While large latent capacity can be beneficial, it must be paired with sustained flattening mechanisms to prevent top-heavy eigenspectrum from re-emerging.

3.4 Impact of LayerNorm Positioning on the FFN Latent Space Dimensionality

PreLN turns width into usable dimensions. Across the FFN-width sweep, the normalized PR, which reflects the effective utilization of available latent space, for PreLN is highest and remains nearly flat as D increases. Figure 6 shows the layer-consistent behavior for PreLN, highlighting the conversion of added width into usable dimensions. In effect, PreLN offers the best return-on-width.

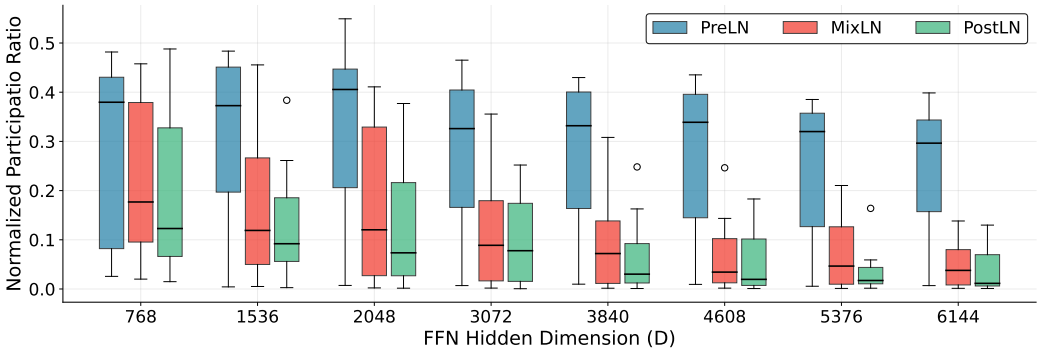

Figure 6: LayerNorm positioning and FFN width sweep: We report participation ratio normalized by hidden size D for PreLN, MixLN, and PostLN configurations. PreLN sustains the highest and most stable utilization of FFN width across the sweep, PostLN incurs diminishing return at higher FFN width, MixLN lies in between but with greater layer-to-layer variability.

PostLN shows diminishing returns at higher width. The width utilization is lowest for PostLN and *decreases with* D, revealing growing spectral concentration—added capacity is concentrated into fewer dominant directions instead of broadening the effective dimensionality. MixLN is intermediate with medians between PreLN and PostLN and wider layer-to-layer spread, implying a less stable inductive bias across depth. Thus, LayerNorm placement governs how width is spent. Table 4

shows the raw metric values. For a detailed discussion on spectral signatures of these normalization techniques (Figure 8), refer to Appendix C.1.

3.5 Layerwise Dynamics for Positional Encoding: RoPE vs NoPE

RoPE prevents mid-to-deep spectral collapse, improving depth utilization

Figure 7 demonstrate that the NoPE's PR declines in the middle and deeper layers, indicating that representations collapse into a narrow subspace and *squander* model depth. RoPE. on the other hand, sustains higher PR across the mid-to-deeper layers, improving the depth utilization. This effect aligns with recent evidence that intermediate layers are disproportionately important—a sweet spot between compression and preservation **?**, and a critical feature engineering and ensembling phases of computation Lad et al. (2024)—which collapse under NoPE but remain effective under RoPE. These improved spectral utilization in RoPE helps achieves lower evaluation perplexity (15.20 vs 16.78) than NoPE. Figure 9 in Appendix shows the spectral entropy heatmaps reaffirming that RoPE prevents the mid-to-deep spectral collapse characteristic of NoPE.

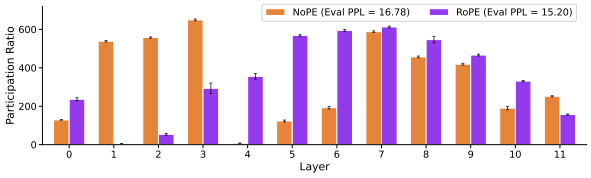

Figure 7: Layerwise participation ratio (PR) comparison (RoPE vs NoPE) in GPT-2 models trained from scratch on 26B token form openwebtext dataset with 512 context length for 100K steps. RoPE sustains higher PR in the middle and deeper layers, indicating better utilization of latent space and network's depth.

4 Related Work

Spectral diagnostics in deep neural networks. Prior work uses spectral signals primarily on representations or weights. RankMe Garrido et al. (2023) and Diff-eRank Wei et al. (2024) apply spectral-entropy-based Rank measures for hidden states to predict downstream accuracy and quantify compression, respectively. Bao et al. (2024) established the link between spectral concentration of QK weight matrix and attention localization, which Lee et al. (2025) addressed using one-step belief-propagation refinement. Hu et al. (2025) showed that weight ESD heavytailness is biased by layer aspect ratio and propose fixed-aspect sub-ESD averaging to debias, and Hu et al. (2025) analyzes layerwise representations using matrix/spectral entropy to select strong mid-depth embeddings.

In contrast to rank-based representation proxies, attention-weight analyses, or weight-ESD debiasing, our approach directly explains architectural effects within FFNs through eigenspectrum dynamics.

5 Computational Complexity and Overheads of NerVE

To enable scalable eigenvalue analysis during training, we implemented memory optimization strategies, listed in Algorithm 1 (Appendix F.1). As shown in Table 2, these memory optimization significantly reduces the peak GPU memory usage. For instance, for GPT-2 model (D=4d) the peak GPU memory usage is restricted to $\approx 2 \times 36$MB per layer rather than accumulating $2 \times 12 \times 36$MB = 864MB across all FFNs per logging step. For absolute wall-clock time, refer to Table 6.

Table 2: GPU memory overhead for full-batch eigen-computation across various FFN widths.

Metric	D=1d	D=2d	D=3d	D=4d	D=5d	D=6d	D=7d	D=8d
Matrix dim.	[768,768]	[1536, 1536]	[2304, 2304]	[3072, 3072]	[3840, 3840]	[4608, 4608]	[5376, 5376]	[6144, 6144]
Matrix size	2.25MB	9MB	20.25MB	36MB	56.25MB	81MB	110.25MB	144MB
GPU Mem.	4.5MB	18MB	40.5MB	72MB	112.5MB	162MB	220.5MB	288MB

6 Limitations and Conclusion

While our four framework analyze how FFNs organize variance in LLMs, they do not directly predict downstream task quality. Moreover, computing full eigendecompositions in large dimensions can be costly, often necessitating sampling or approximation. Despite these constraints, our analysis shows that each metric contributes a distinct and complementary view of high-dimensional usage, revealing top-heaviness, effective rank, early enrichment, and distribution shifts.

REFERENCES

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Han Bao, Ryuichiro Hataya, and Ryo Karakida. Self-attention networks localize when QK-eigenspectrum concentrates. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Manlio De Domenico and Jacob Biamonte. Spectral entropies as information-theoretic tools for complex network comparison. *Physical Review X*, 2016.

Harm de Vries. In the long (context) run: It's not the quadratic attention; it's the lack of long pre-training data. `https://www.harmdevries.com/post/context-length/`, 2023.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, 2017.

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 (ICML)*, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Bobby He and Thomas Hofmann. Simplifying transformer blocks. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Yu Hu and Haim Sompolinsky. The spectrum of covariance matrices of randomly connected recurrent neuronal networks with linear dynamics. *PLoS computational biology*, 2022.

Yuanzhe Hu, Kinshuk Goel, Vlad Killiakov, and Yaoqing Yang. Eigenspectrum analysis of neural networks without aspect ratio bias. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

HuggingFace. Codeparrot. `https://huggingface.co/learn/nlp-course/chapter7/6`.

Nandan Kumar Jha and Brandon Reagen. ReLU's revival: On the entropic overload in normalization-free large language models. *NeurIPS Workshop on Attributing Model Behavior at Scale*, 2024.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Analyzing feed-forward blocks in transformers through the lens of attention map. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

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

Nakyung Lee, Yeongoon Kim, Minhae Oh, Suhwan Kim, Jin Woo Koo, Hyewon Jo, and Jungwoo Lee. Mitigating attention localization in small scale: Self-attention refinement via one-step belief propagation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-LN: Unleashing the power of deeper layers by combining pre-LN and post-LN. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 1991.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *Advances in neural information processing systems*, 2017.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Anna Marbut, Katy McKinney-Bock, and Travis Wheeler. Reliable measures of spread in high dimensional latent spaces. In *International Conference on Machine Learning (ICML)*, 2023.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Alexander V Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. Kernel language entropy: Fine-grained uncertainty quantification for LLMs from semantic similarities. In *The 38th Annual Conference on Neural Information Processing Systems(NeurIPS)*, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of machine learning research (JMLR)*, 2020.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, 2016.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. In *Neurocomputing*, 2024.

Lai Wei, Zhiquan Tan, Chenghai Li, Jindong Wang, and Weiran Huang. Diff-erank: A novel rank-based metric for evaluating large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *5th ICLR Workshop on practical ML for limited/low resource settings (PML4LRS)*, 2024.

A Design of Experiments and Implementation Details

Implementation Details for Computing Covariance Matrices

Our implementation ensures numerical stability and efficiency through:

1. *Precision control*: Converting all tensors to float32 to avoid precision issues.
2. *Numerical stability*: Adding small epsilon values (1e-12) to prevent division by zero.
3. *Memory efficiency*: For distributed training, gathering activations from all GPUs to rank 0.
4. *Specialized computation*: Using `torch.linalg.eigvalsh` for symmetric matrices.

During training, activations are collected through registered PyTorch hooks. For pre-activation collection, we use forward hooks on the output of the up-projection layer. For post-activation collection, we use pre-forward hooks on the down-projection layer to capture inputs before they enter the down-projection.

B Eigenvalue Distribution Sensitivity of Spectral Metrics

SE characterizes the shape of the spectrum by computing a normalized entropy over eigenvalues, and it is *highly* sensitive even to the smaller eigenvalues due to the logarithmic weighting of $\lambda_i / \sum_j \lambda_j$. As a result, SE increases with broader tails and offers a fine-grained view of how variance is distributed, especially in the mid-to-lower spectrum. Participation ratio, by contrast, *suppresses* the influence of smaller eigenvalues due to insignificant contribution to $\sum_i \lambda_i^2$. Consequently, PR is less sensitive to gradual slope changes among the top eigenvalues compared to SE.

While SE and PR consider the *entire* set of eigenvalues in different ways, EEE targets the front-loadedness of the spectrum. Hence, EEE is *most* sensitive to front-loaded spectrum where a handful of large eigenvalues drives EEE close to 1, even if the rest are moderate.

Meanwhile, JS divergence plays a distinct role and compares two different distribution (e.g. pre- vs. post-activation) in shape rather than magnitude. In that sense, JS captures a distinct aspect that SE, PR, and EEE do not: while the first three measure intrinsic properties of a single distribution, JS quantifies the *information-theoretic distance* between two distributions.

Notably, all four metrics are invariant to uniform scaling of the eigenvalues $\{\lambda_i\}_{i=1}^{D}$. This is crucial in practice, as the magnitude of the covariance matrix may fluctuate due to various factors such as changes in batch statistics. Scale invariance ensures the metrics remain focused on the shape of the distribution, which truly governs the directionality in the latent space. Moreover, each metric accounts for the entire eigenvalue distribution, unlike simple measures such as the eigenvalue ratio (largest-to-smallest), which are sensitive only to extremes.

C Eigenspectral Signature

C.1 Spectral Signature of LayerNorm Positioning: PreLN, MixLN, and PostLN

Figure 8 illustrates the post-activation spectral signatures (SE_post and PR_post) for three LayerNorm placements—PreLN, PostLN, and MixLN—across GPT2-125M, LLaMA-70M, and LLaMA-130M. These signatures highlight the extent of FFN latent space utilization across layers. Within each model family, the ranking of evaluation perplexities is corroborated by their spectral signatures: models with lower perplexity exhibit higher utilization.

GPT2-125M: Performance follows the order PreLN (PPL = 2.714) $>$ MixLN (2.808) $>$ PostLN (2.830). The spectral signatures follow this ranking: PreLN exhibits superior spectral entropy and participation ratio trend across layers, indicating more effective FFN latent space utilization, while PostLN shows the most constrained spectral characteristics. In particular, $L7$ in PostLN and MixLN show very-low utilization compared to the PreLN configuration.

LLaMA-70M: PostLN achieves the lowest perplexity (PPL = 33.6), followed by MixLN (33.9), while PreLN performs worst (34.2). The eigenspectral analysis shows that PreLN exhibits substantially lower spectral entropy in deeper layers ($L7$-$L8$) compared to PostLN and MixLN. In contrast,

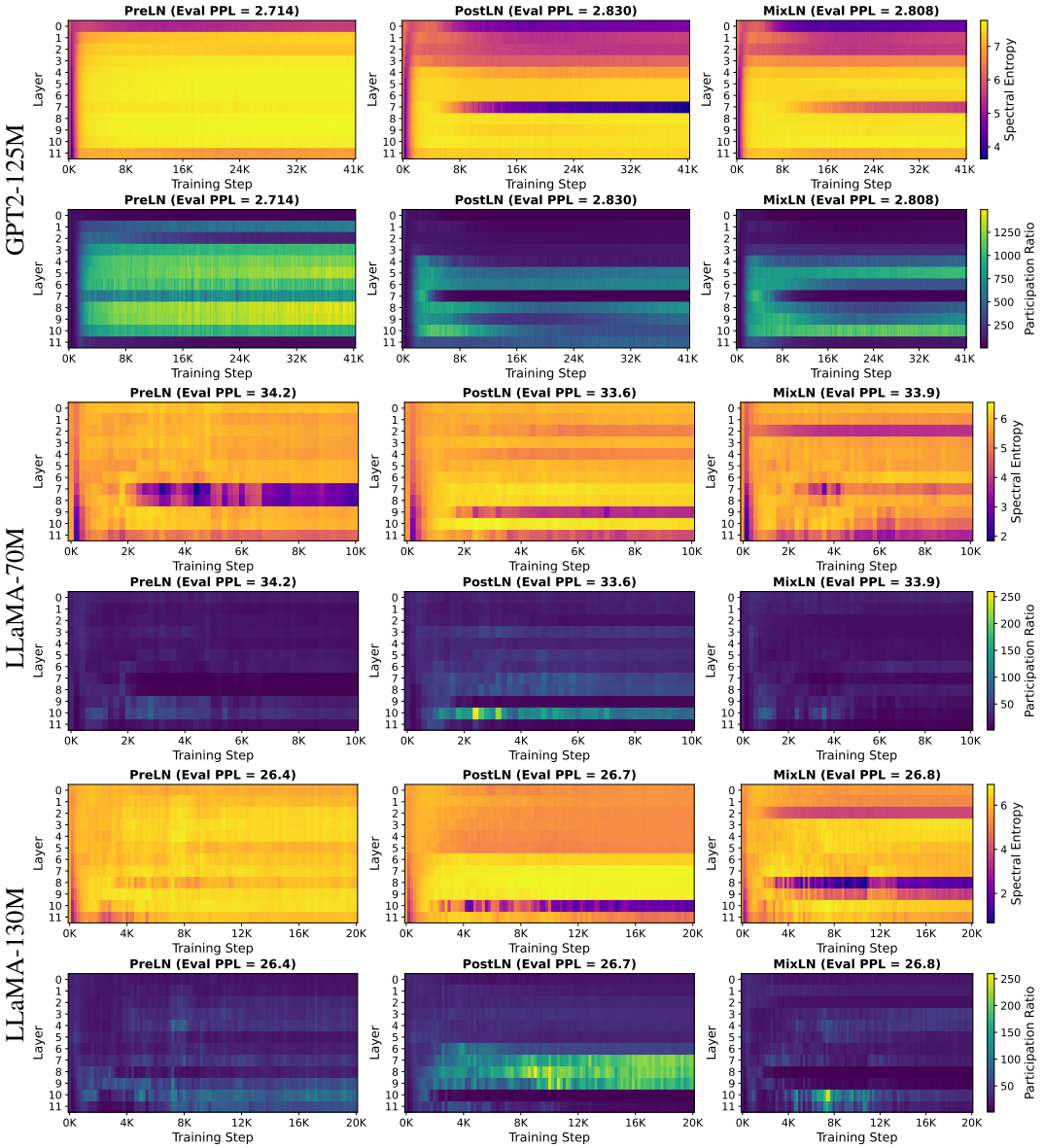

Figure 8: Eigenspectral impact of LayerNorm placement (PreLN, PostLN, MixLN) in GPT-2 and LLaMA variants (70M, 130M). The spectral signatures are shown through post-activation spectral entropy (↑) and participation ratio (↑), and model's perplexity is shown on top of each plots. GPT-2 models trained on CodeParrot and LLaMA variants on C4.

PostLN demonstrates superior spectral characteristics with higher participation ratios in deeper layers compared to MixLN, consistent with its lower perplexity.

LLaMA-130M: PreLN yields the best performance with a perplexity of 26.4, followed closely by PostLN (26.7) and MixLN (26.8). While PostLN exhibits stronger spectral entropy and participation ratio in the mid-depth layers ($L6$-$L9$), PreLN consistently outperforms across the remaining layers, particularly $L10$ where PostLN deteriorates, undermining its overall benefits. In contrast, MixLN shows the worse spectral profile, leads to highest perplexity.

These results suggest that LayerNorm placement significantly influences how effectively the FFN utilizes its latent space. Pre-LN configurations appear to better preserve and amplify feature diversity, especially in deeper layers, thereby enabling higher-dimensional latent representations. The observed gains in the MixLN setup indicate that even partial use of PreLN can compensate for the limitations of PostLN, offering a potential strategy for balancing stability and expressivity.

C.2 Spectral Signature of Positional Encoding: NoPE vs RoPE

Figure 9 shows the spectral signature of rotatory positional encoding (RoPE), in contrast with no positional encoding (NoPE). In particular, the layerwise spectral entropy and EEE values of post-activation eigenspectrum are shown.

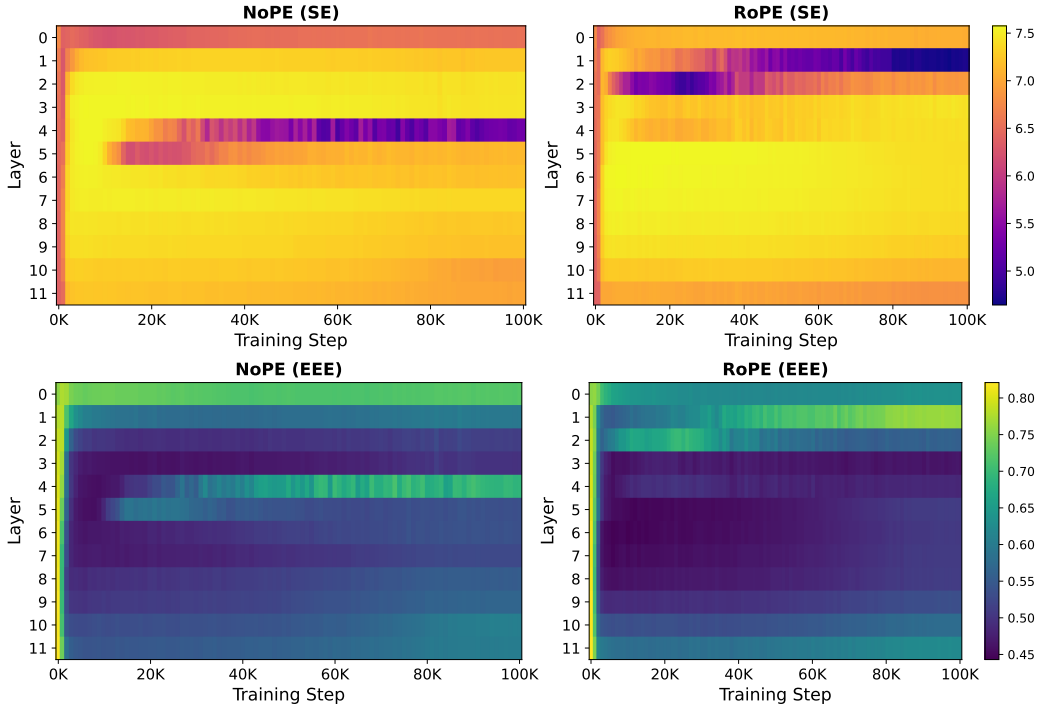

Figure 9: Spectral Signature of Positional Encoding (RoPE vs NoPE) in GPT-2 models trained with 512 context size on 26B token form openwebtext dataset

D Additional Results

Table 3: Correlation between validation loss/perplexity and eigen-metrics (SE and PR). Left: Across FFN width configurations in GPT-2 (GELU) models. Right: Across architectures and activation functions. Higher SE/PR implies lower loss regardless of configuration.

Metric	FFN Width Configuration (GPT-2 GELU)								GPT-2				NormFree GPT-2			LLaMA
	D=1d	D=2d	D=3d	D=4d	D=5d	D=6d	D=7d	D=8d	GELU	ReLU	GeGLU	SwiGLU	GELU	ReLU	LReLU	71M
SE_pre	-0.98	-0.98	-0.99	-0.99	-0.99	-0.99	-0.99	-0.99	-0.99	-0.98	-0.95	-0.97	-0.82	0.03	0.03	-0.96
SE_post	-0.84	-0.84	-0.86	-0.87	-0.87	-0.87	-0.87	-0.87	-1.00	-1.00	-0.57	-0.85	-0.92	-0.99	-1.00	-0.87
PR_pre	-0.97	-0.98	-0.98	-0.99	-0.98	-0.97	-0.98	-0.97	-0.99	-0.98	-0.97	-0.97	-0.93	-0.55	-0.60	-0.84
PR_post	-0.85	-0.93	-0.94	-0.94	-0.95	-0.95	-0.93	-0.93	-1.00	-0.97	-0.94	-0.89	-0.99	-0.94	-0.99	-0.62

Table 4: Raw Metrics - Final Values (median ± MAD across 12 layers)

Method	Metric	D=768	D=1536	D=2048	D=3072	D=3840	D=4608	D=5376	D=6144
PreLN	PR	292±62	573±149	831±238	1002±354	1275±333	1562±429	1721±244	1822±370
	Exp(SE)	546±39	1109±80	1631±91	2154±146	2678±158	3185±200	3732±133	4169±175
MixLN	PR	137±111	184±126	247±233	274±224	278±234	159±142	251±201	233±187
	Exp(SE)	434±123	814±214	1255±336	1450±628	1907±525	1941±607	2108±923	2229±962
PostLN	PR	95±65	142±104	151±147	240±231	117±110	91±83	94±84	71±62
	Exp(SE)	374±170	815±233	1008±485	1361±613	1064±863	1225±1040	828±619	1148±1010

Table 5: Effect of Spectrum Approximations (sub-sampling and low-rank approximation) on correlation dimensionality measure and eval loss in GPT-2. Token-level sub-sampling preserves pre-metric trends but degrades post-metric correlations because tail eigenvalues are under-sampled; low-rank truncation distorts both.

Metric	Sampling				RandSVD		Lanczos	
	5%	10%	25%	50%	256	512	256	512
SE_pre	-0.97	-0.972	-0.971	-0.972	-0.106	-0.174	-0.822	-0.959
SE_post	-0.34	-0.376	-0.332	-0.363	0.612	0.568	0.335	0.08
PR_pre	-0.914	-0.915	-0.918	-0.912	0.014	0.047	-0.788	-0.901
PR_post	-0.122	-0.138	0.049	-0.235	0.594	0.575	0.314	0.136

Table 6: **Computational Overhead:** Wall-clock time and relative overhead for computing eigenspectrum metrics at various logging frequencies. Overhead is reported as percentage of total training time when eigendecomposition is performed every 200 and 1000 steps on GPT-2 with 3072×3072 FFN covariance matrix size running on AMD EPYC 7502 server with NVIDIA RTX 3090 GPU

Metric	Sampling				RandSVD		Lanczos		Full
	5%	10%	25%	50%	256	512	256	512	batch
Wall-clock time	9.89s	9.92s	11.48s	12.62s	10.28s	11.19s	12.07s	12.20s	14.41s
Overhead-200 (%)	4.37	4.39	5.08	5.58	4.55	4.95	5.34	5.40	6.38
Overhead-1K (%)	0.87	0.88	1.02	1.12	0.91	0.99	1.07	1.08	1.28

E SPECTRAL SIGNATURE ACROSS FFN WIDTH SWEEPS

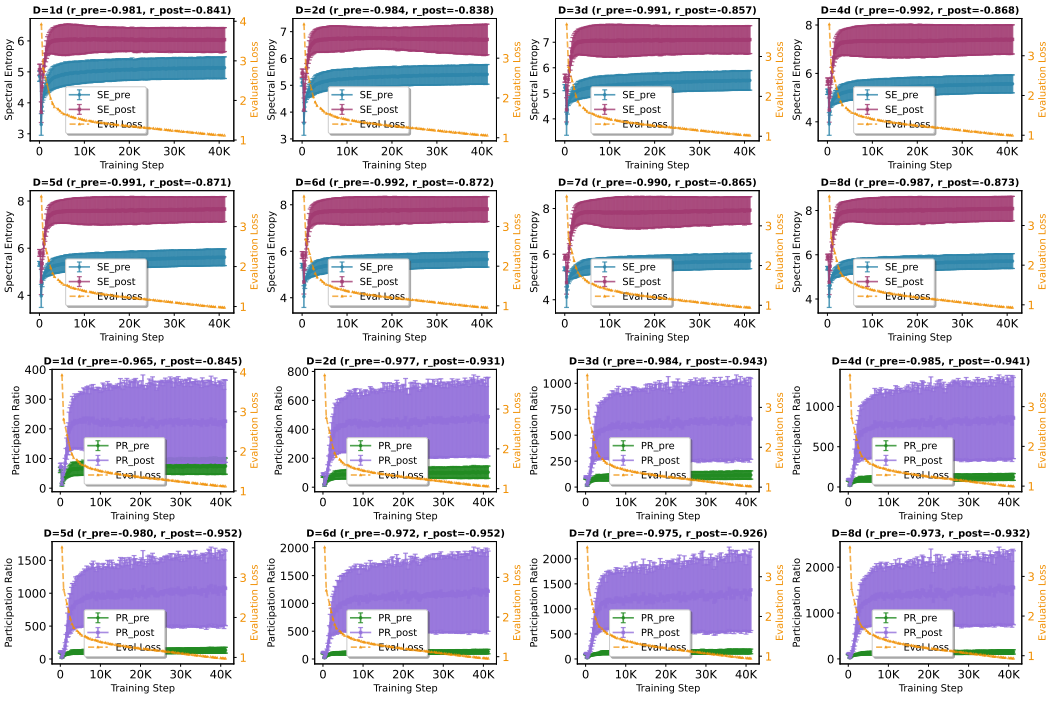

Figure 10: Eigen metrics in GPT-2 (**GELU**, D=1d to 8d) with Pearson r to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

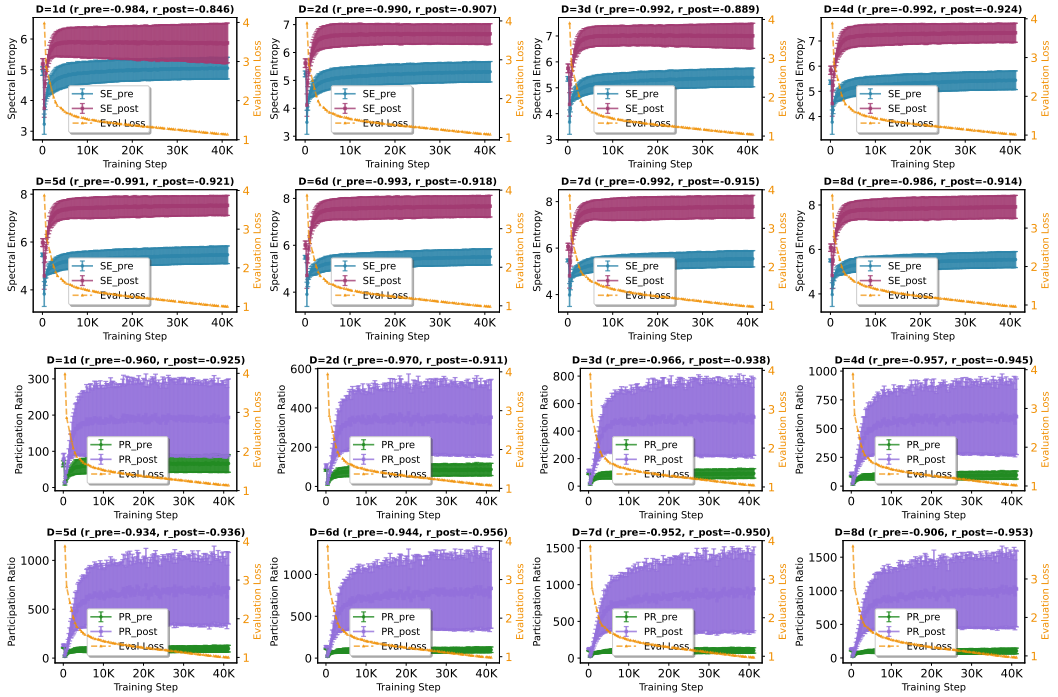

Figure 11: Eigen metrics in GPT-2 (**ReLU**, D=1d to 8d) with Pearson r to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

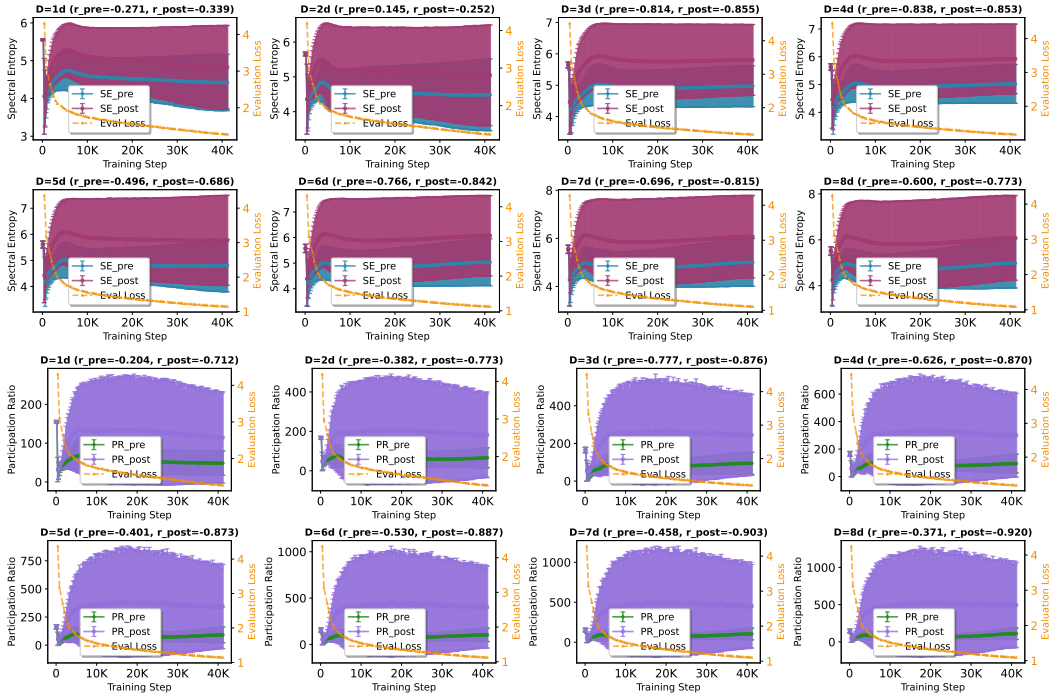

Figure 12: SE and PR in **Normalization-free** GPT-2 (**GELU**, D=1d to 8d) with Pearson correlation to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

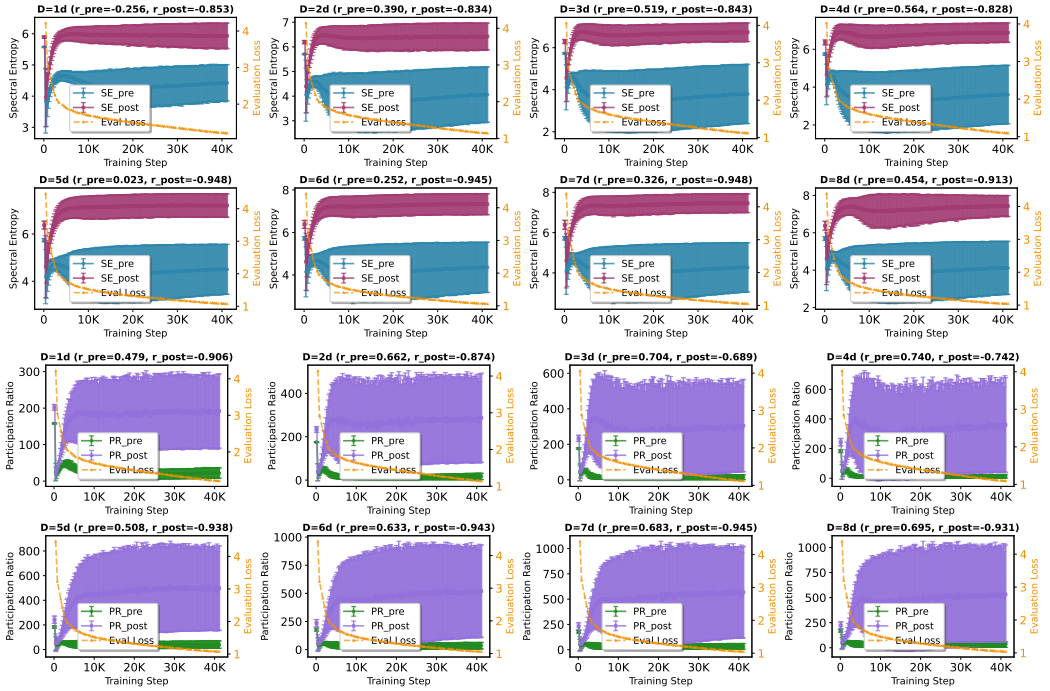

Figure 13: SE and PR in **Normalization-free** GPT-2 (**ReLU**, D=1d to 8d) with Pearson correlation to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

F Spectral Signature of Sampling and Low Rank Approximation

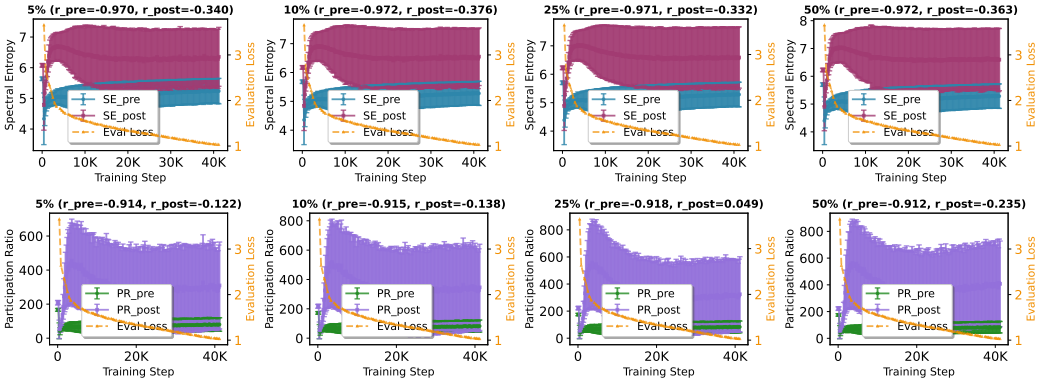

Figure 14: SE and PR in GPT-2 with Pearson r to eval loss under **sub-sampling** (5, 10, 25, 50%)

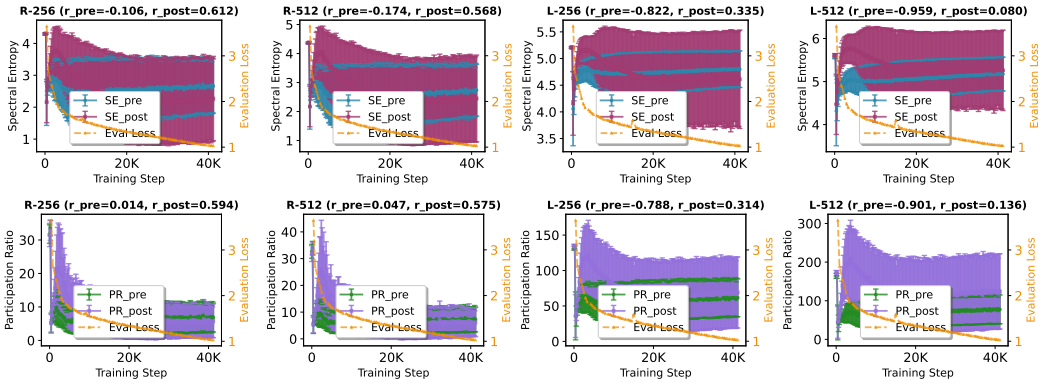

Figure 15: SE and PR in GPT-2 with Pearson r to eval loss under **low-rank approximation**.

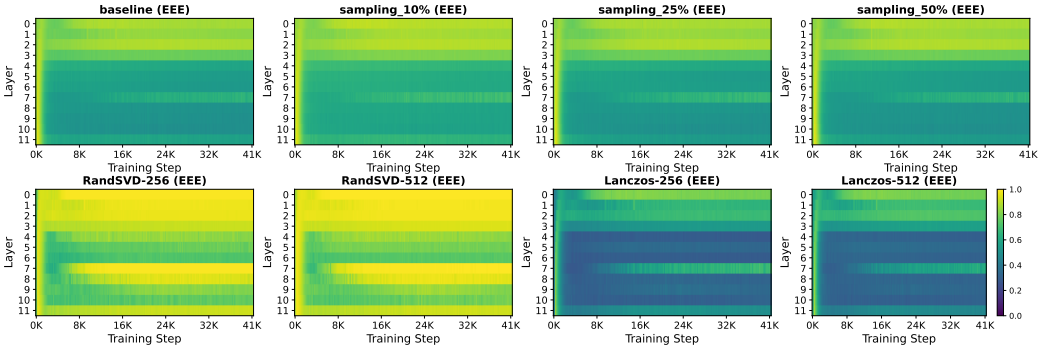

Figure 16: Impact of sampling (10%, 25%, 50%), and low-rank approximation on EEE eigenmetric in GPT-2

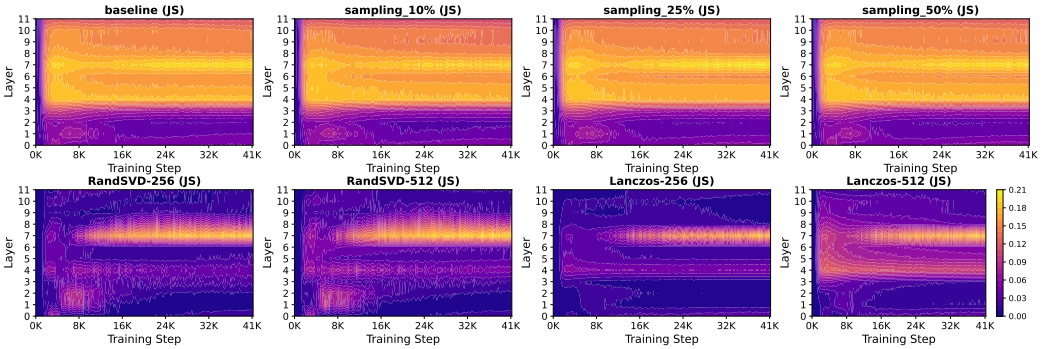

Figure 17: Impact of sampling (10%, 25%, 50%), and low-rank approximation on JS divergence eigenmetric in GPT-2

```
# Start timing (includes all operations below)
torch.cuda.synchronize()
start_time = time.time()

# Complete eigenvalue pipeline for all layers:
# 1. CPU-GPU data transfer of activations
# 2. Covariance matrix computation (3072x3072)
# 3. Eigenvalue decomposition using torch.linalg.eigvals()
# 4. Eigenvalue metrics computation (SE, PR, EEE, JS)
# 5. Memory cleanup and GPU cache clearing

torch.cuda.synchronize()
end_time = time.time()
overhead = end_time - start_time  # Complete wall-clock time
```

Listing 1: Wall-clock timing measurement for eigenspectrum-based computational overhead

Algorithm 1 Memory-Efficient Eigenspectrum Analysis

for layer $\ell = 1, \ldots, L$ **do**
  $\mathbf{H}_\ell \leftarrow \text{GetActivations}(\ell)$ ▷ Store on CPU
  $\boldsymbol{\lambda}_\ell \leftarrow \text{eigvalsh}(\mathbf{H}_\ell^\top \mathbf{H}_\ell / N)$ ▷ O(d) memory
  Compute metrics: SE, PR, EEE, JS from $\boldsymbol{\lambda}_\ell$
  del $\mathbf{H}_\ell$ ▷ Immediate cleanup
end for

F.1 Memory-efficient eigenspectrum analysis

To enable scalable eigenvalue analysis during training, we implement three memory optimization strategies that significantly reduce GPU memory overhead:

1. **Eigenvalue-only computation:** Since our eigen metrics depend only on eigenvalues, we employ `torch.linalg.eigvalsh` to compute eigenvalues without eigenvectors.

```
### What we're using (efficient):
# Only eigenvalues: 3072 values
vals = torch.linalg.eigvalsh(cov)

### Less efficient alternatives:
# Eigenvalues + eigenvectors: 3072 + (3072x3072)
vals, vecs = torch.linalg.eigh(cov)
# General eigendecomposition (even more overhead)
vals, vecs = torch.linalg.eig(cov)
```

Listing 2: Memory-efficient eigenvalue computation

2. **Sequential layer processing:** Rather than computing eigenvalues for all layers simultaneously, we process layers sequentially with memory cleanup between computations:

```
# Process each layer individually with cleanup
for layer_idx in sorted(self.layer_pre_acts.keys()):
    # Compute metrics for current layer
    # ...
    self.layer_pre_acts[layer_idx].clear()
    gc.collect()
    torch.cuda.empty_cache()
```

Listing 3: Sequential layer processing with memory cleanup

   This approach maintains peak GPU memory usage at $\sim$2$\times$36MB per layer rather than accumulating 2$\times$12$\times$36MB = 864MB across all FFNs per logging step (Table 2).
3. **Hybrid storage strategy:** We store activation tensors on CPU memory while performing eigenvalue computations on GPU, balancing memory efficiency with computational speed.