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, and that optimizer geometry strongly modulates the extent of this variance reinjection. 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; and optimizer choices redistributing effective capacity across depth. 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 and optimizer design 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).

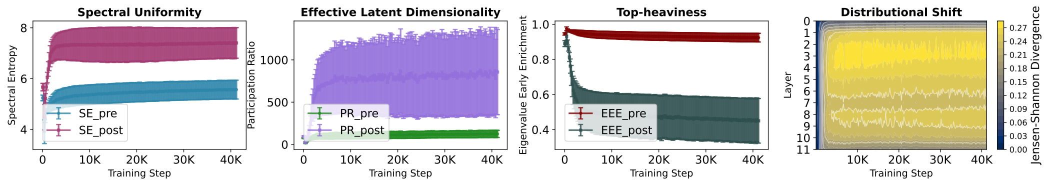

Figure 1: Nonlinear eigenspectrum dynamics in FFNs (GPT-2): FFN nonlinearity (GELU) regulate information flow by reinjecting variance, reactivating under-utilized directions (post-activation SE↑ and PR↑), and flattening the eigenspectrum, less top-heavy (post-activation EEE↓), in the high-dimensional 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.

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 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); positional encoding scheme (Su et al., 2024); and the optimizer choices, including AdamW (Loshchilov & Hutter, 2019), Muon (Jordan et al., 2024), and Dion (Ahn et al., 2025).

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, OpenWebText, FineWeb (Penedo et al., 2024), and C4 datasets, 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:

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

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; Huang et al., 2023).

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,

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

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):

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

EEE ≈ 1 indicates that most of the variance is concentrated in the top few directions, forming a steep eigenvalue spectrum; conversely, EEE ≈ 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.

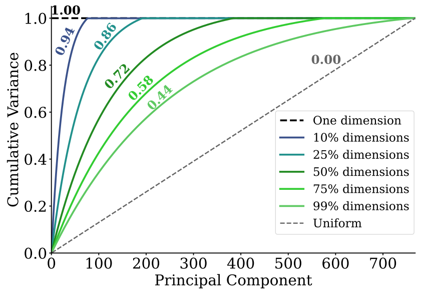

Figure 2: Cumulative variance distribution across a 768-dimensional latent space. Higher values (shown on curves) indicates top-heavy concentration in a few dominant directions, while lower one reflects a more uniform distribution.

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)):

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

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

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

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

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

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 (Briët & Harremoës, 2009), 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 (Dong et al., 2025; Nielsen, 2025), 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 distributional sensitivity of 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 (Raffel et al., 2020), 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). To study optimizer-dependent (AdamW, Muon, Dion) dynamics, we train GPT-2 350M and 160M variants on FineWeb (Penedo et al., 2024) dataset.

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, LLaMA-250M, and LLaMA-1.3B variants are trained on 2.2B tokens for 20K steps. All LLaMA models use a context length of 256. For optimizer-specific eigenspectrum analysis, we train GPT-2 models with 512 and 1024 context lengths, following the hyperparameter settings from Ahn et al. (2025).

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.

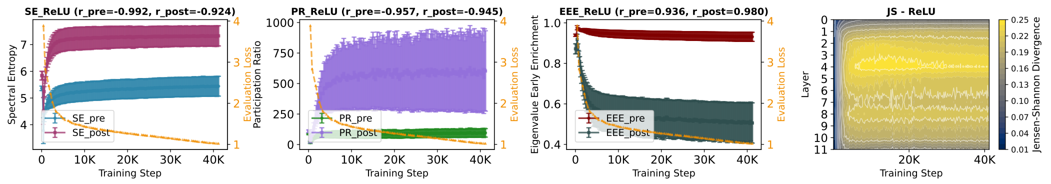

Figure 3: Eigenspectrum dynamics illustrate *how FFN nonlinearities regulate information flow and reshape the eigenspectrum* during training for GPT-2 (ReLU) on CodeParrot. Pre- and post-activation dynamics are shown for SE, PR, and EEE, highlighting how nonlinearities reinject variance and alter spectral 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.

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 pushes PR$_{\text{post}}$ 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 EEE$_{\text{post}}$ $\approx$ 1 and JS $\approx$ 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$_{\text{post}}$ 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

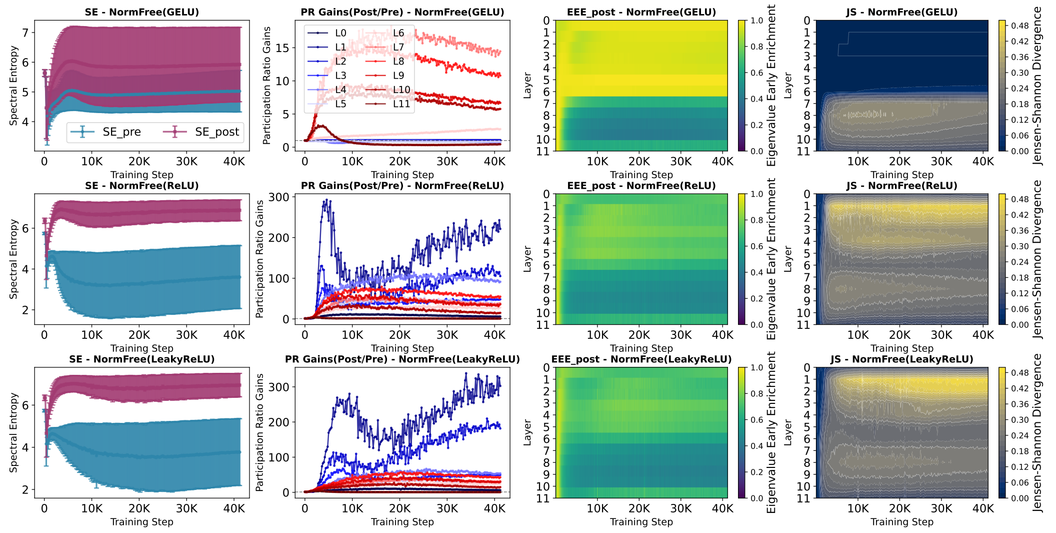

Figure 4: Eigenspectrum dynamics for norm-free GPT-2 models using GELU (top), ReLU (middle), and learnable-slope Leaky ReLU (bottom). Columns show layer-averaged SE (pre vs. post), PR gain (post to pre), post-activation EEE (yellow $\rightarrow$ top-heavy), and JS (yellow $\rightarrow$ strong redistribution) 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 $> \mathbf{200}\times$) and flattening the spectrum (EEE < 0.3).

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 ≈50% (refer to Table 1).

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 ↓) 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 (≈-0.45) among all parametric normalization methods, indicating balanced variance distribution across a moderate number of directions (PR_post ≈ 200) and improved latent space utilization. In contrast, WNorm shows delayed and

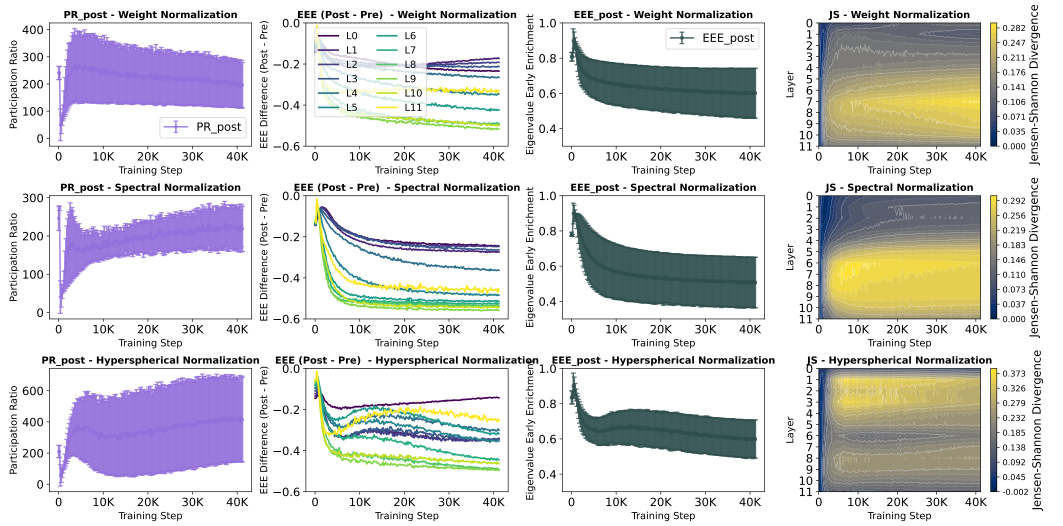

Figure 5: Impact of FFN (parametric) normalization in norm-free GPT-2 with learnable-slope leaky ReLU. Eigenspectrum dynamics are quantified by latent capacity (PR_post), spectral regularization and flattening (ΔEEE and EEE_post), and distributional shift (JS). Top to bottom: Weight, Spectral, and Hyperspherical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

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 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 while PostLN shows diminishing returns at higher width. 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.

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 10), refer to Appendix C.1.

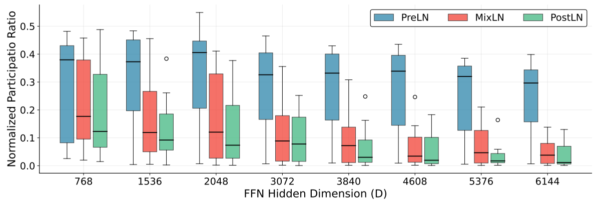

Figure 6: LayerNorm positioning and FFN width sweep: Post-activation participation ratio is normalized by D for PreLN, MixLN, and PostLN configurations. PreLN sustains the highest and most stable utilization of FFN width across the sweep, PostLN incurs diminishing return at higher FFN width, MixLN lies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

3.5 Layerwise Dynamics for Positional Encoding: RoPE vs NoPE

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

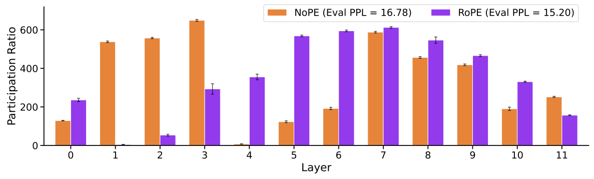

Figure 7: Layerwise participation ratio (PR) comparison (RoPE vs NoPE) in GPT-2 models trained from scratch on 26B token form openwebtext dataset with 512 context length for 100K steps. RoPE sustains higher PR in the middle and deeper layers, indicating better utilization of latent space and network's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

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 (Skean et al., 2025), 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 12 in Appendix shows the spectral entropy heatmaps reaffirming that RoPE prevents the mid-to-deep spectral collapse characteristic of NoPE.

3.6 Optimizer-Dependent Role of FFN Nonlinearity: Repair vs Refinement

To understand the optimizer-dependent role of FFN nonlinearity, we examine eigenspectrum dynamics under three LLM optimizers; AdamW, Muon, and Dion; and summarize the observations below.

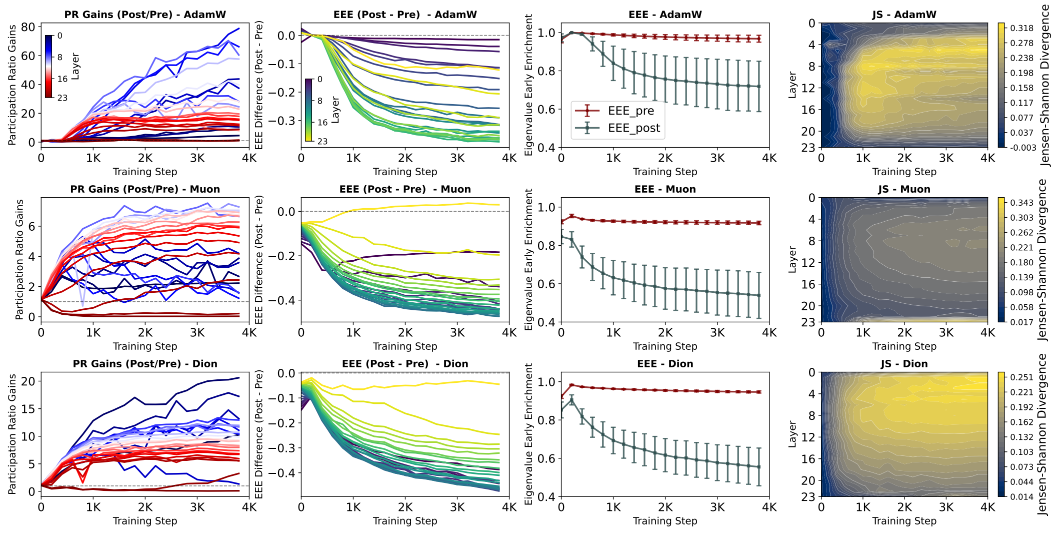

Figure 8: Optimizer-dependent FFN eigenspectrum dynamics in GPT2-350M. Rows show AdamW (top), Muon (middle), and Dion (bottom). AdamW has large early PR gains and high JS with relatively high EEE_post, indicating optimizer-induced pre-activation collapse followed by aggressive but incomplete nonlinear repair. Muon shows the smallest PR gains, lowest JS, and lowest EEE_post, and flatter post-spectra. Dion is intermediate and falls between these two regimes, improving over AdamW but not matching Muon's pre-/post-spectral behavior. The perplexity ordering (Muon > Dion > AdamW) aligns with post-activation spectral flatness.

Muon minimizes nonlinearity burden by preserving activation-compatible eigenspectrum Figure 8 shows that Muon maintains uniformly small PR(Post/Pre) gains and consistently low JS divergence throughout training. This combination indicates that Muon keeps the pre-activation FFN eigenspectra high-dimensional and closely aligned with the activation's output, so the nonlinearity does not need to substantially restructure representations. Dion follows the same trend but less strongly: PR gains are moderate and JS is higher and more uniform across layers, suggesting broader activation-driven reshaping than Muon yet still far less mismatch than AdamW. Thus, geometric updates serves as spectral equalizers by preventing the pre-activation spectrum from drifting into regimes that demand large nonlinear correction.

The early-layer spectral collapse in AdamW forces FFN nonlinearity into repair mode.

In contrast, AdamW exhibits very large PR(Post/Pre) gains concentrated in early layers, indicating optimizer-induced pre-activation collapse, as energy concentrating into a small set of dominant eigenmodes followed by strong nonlinear repair (Figure 9). However, this repair does not translate into better utilization, as the $\mathrm{PR}_{\mathrm{post}}$ remains lower than Muon and Dion in early and intermediate layers despite their massive PR gains during training. Thus, under AdamW the activation expends capacity primarily to undo collapse rather than refine a healthy spectrum, consistent with AdamW's worse perplexity (see Table 8).

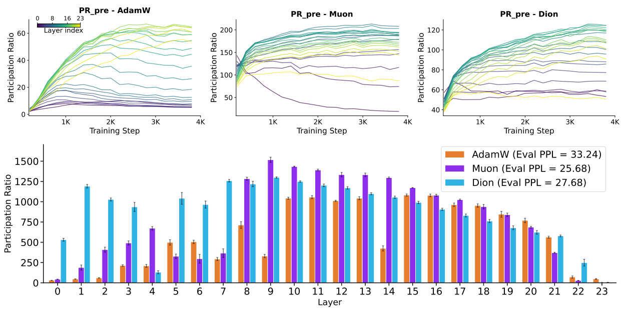

Figure 9: Layerwise PR_pre (top) over training, and final PR_post per-layer (bottom) for AdamW, Muon, and Dion optimizer. Muon maintains the highest PR_pre across almost all layers, Dion is intermediate, and AdamW shows early-layer collapse. Moreover, Muon concentrates the largest effective dimensionality in middle FFNs.

Muon concentrates effective dimensionality where it matters: the middle FFNs. Figure 9 isolates where effective dimensionality ultimately accumulates. Muon achieves the highest $\mathrm{PR}_{\mathrm{post}}$ in the intermediate FFNs; whereas, Dion attains very high $\mathrm{PR}_{\mathrm{post}}$ in the early FFNs, which does not results in the best perplexity; instead, the perplexity ordering tracks $\mathrm{PR}_{\mathrm{post}}$ specifically in the mid block.

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 G). 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 $\approx 2 \times$ 36MB per layer rather than accumulating $2 \times 12 \times$ 36MB = 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

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

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.

Valentyn Boreiko, Zhiqi Bu, and Sheng Zha. Towards understanding of orthogonalization in muon. In *High-dimensional Learning Dynamics 2025*.

Jop Briët and Peter Harremoës. Properties of classical and quantum jensen-shannon divergence. *Physical Review A-Atomic, Molecular, and Optical Physics*, 2009.

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.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning (ICML)*, 2021.

Yuxin Dong, Haoran Guo, Tieliang Gong, Wen Wen, and Chen Li. Exactly tight information-theoretic generalization bounds via binary jensen-shannon divergence. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

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.

Qiyao Huang, Yingyue Zhang, Zhihong Zhang, and Edwin Hancock. ESSEN: Improving evolution state estimation for temporal networks using von neumann entropy. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

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.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. *URL https://kellerjordan. github. io/posts/muon*, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

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 and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

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.

Frank Nielsen. Two tales for a geometric jensen–shannon divergence. *arXiv preprint arXiv:2508.05066*, 2025.

Frank Nielsen and Richard Nock. Total jensen divergences: Definition, properties and clustering. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

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.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in 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. *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.

Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

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

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Advances in neural information processing systems (NeurIPS)*, 2021.

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 (Nielsen & Nock, 2015).

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 10 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,

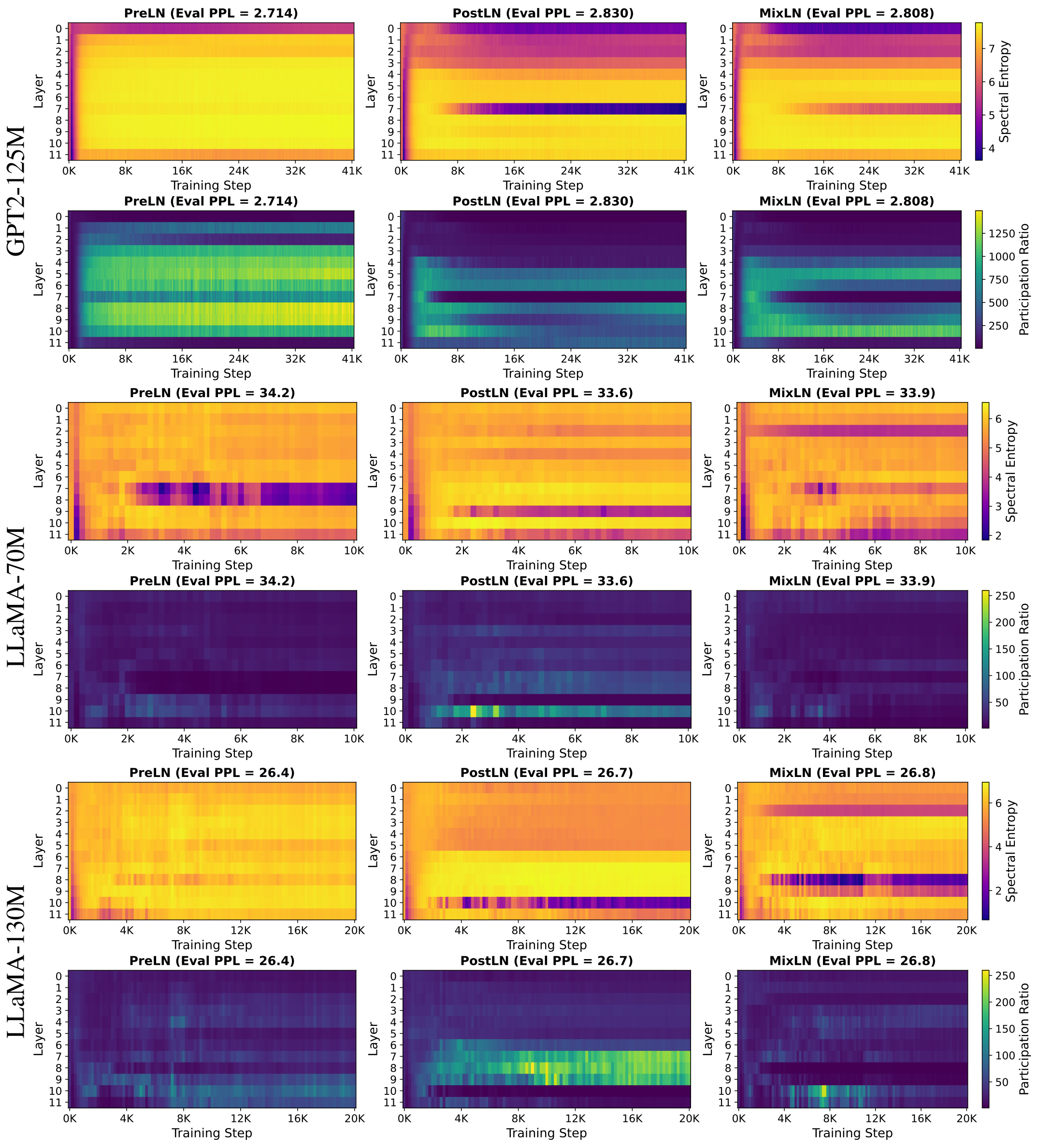

Figure 10: Eigenspectral impact of LayerNorm placement (PreLN, PostLN, MixLN) in GPT-2 and LLaMA variants (70M, 130M). The spectral signatures are shown through post-activation spectral entropy (↑) and participation ratio (↑), and model's perplexity is shown on top of each plots. GPT-2 models trained 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 in Larger LLaMA Models

Figure 11 shows the post-activation eigen-spectral signatures (SE_post and PR_post) for LayerNorm placements, PreLN and MixLN, in LLaMA-250M and LLaMA-1.3B. Note that PostLN is excluded since it becomes unstable at these larger scales (Li et al., 2025). These spectral signatures provide a quantitative assessment for latent space utilization in each FFNs.

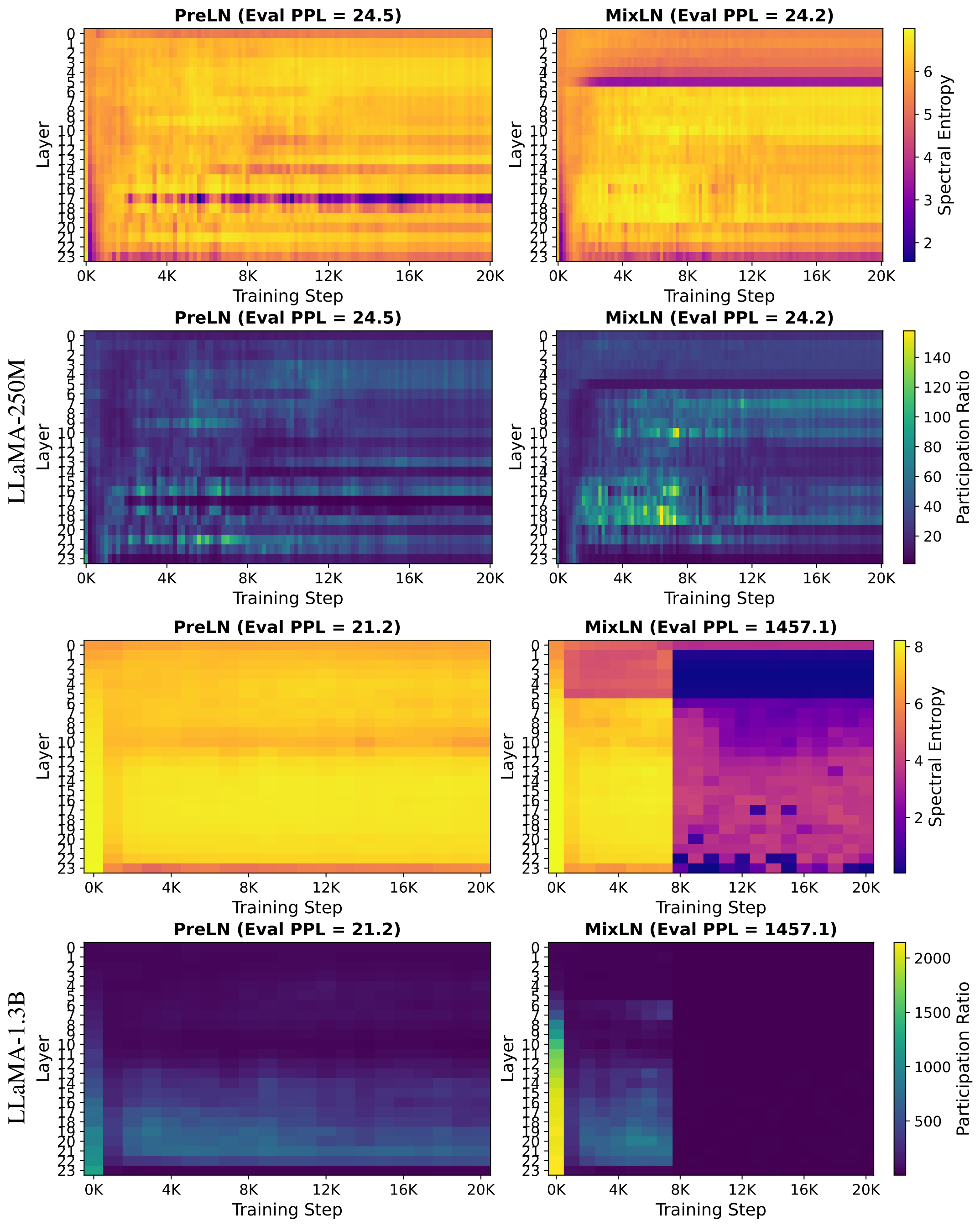

Figure 11: Eigenspectral impact of LayerNorm placement (PreLN vs MixLN) in LLaMA-250 and LLaMA-1.3B models, trained from scratch on C4 dataset for 20K iterations with 256 context length. The spectral signatures are shown through post-activation spectral entropy (↑) and participation ratio (↑), and model's perplexity is shown on top of each plots. MixLN variant of LLaMA-1.3B destabilized after 7K iterations.

LLaMA-250M: MixLN outperforms the PreLN configuration by a margin of 0.3 in terms of PPL improvement (24.2 vs 24.5). The spectral signature of PreLN exhibits a consistent lower SE in layer 7 to 16, and lower PR in mid-depth regions (Figure 11). Whereas, MixLN avoid this mid-network spectral collapse pattern by confining the lower SE bands in early layers while maintaining higher SE and PR through the mid-depth layers, *effectively placing most of the usable capacity where the model does the bulk of its computation*. This redistribution of spectral entropy and participation ratio across depth is consistent with the lower perplexity achieved by the MixLN model.

LLaMA-1.3B: At 1.3B scale, MixLN's strategic LayerNorm placement unable to facilitate favorable capacity distribution, and training collapses after 7K steps. Whereas, PreLN maintain a stable spectral entropy across layers and achieves notably higher participation ratio in the deeper layers throughout training. This divergence in spectral behavior aligns with the final performance gap, as MixLN's perplexity explodes to 1457.1 compared to 21.2 for PreLN.

C.3 Spectral Signature of Positional Encoding: NoPE vs RoPE

Figure 12 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.

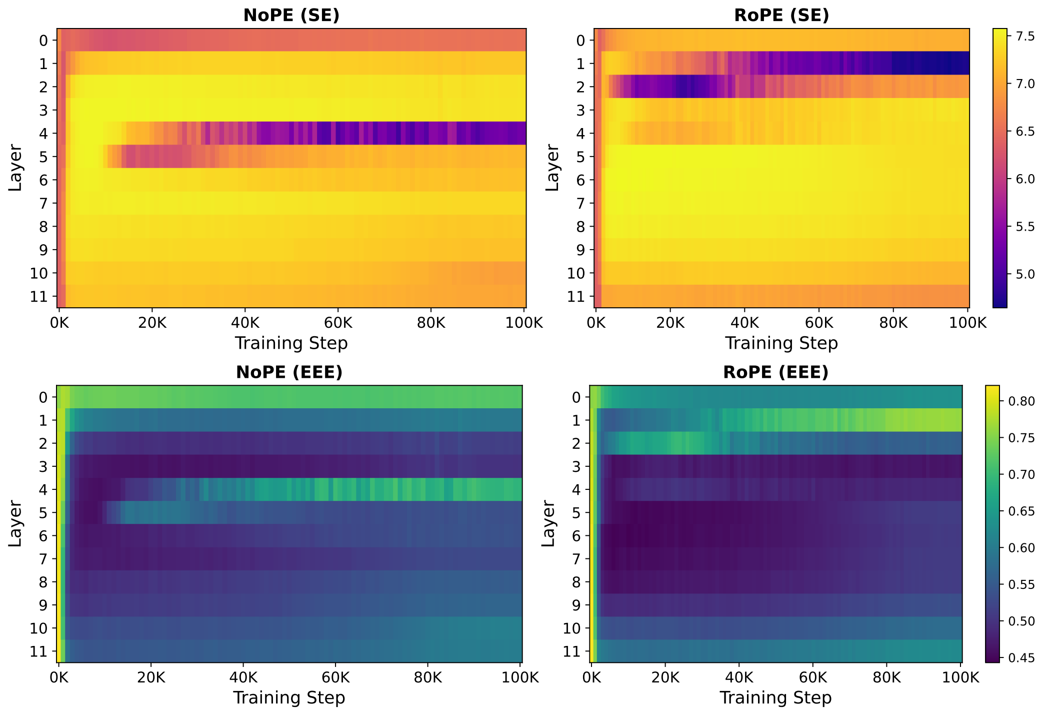

Figure 12: Spectral Signature of Positional Encoding (RoPE vs NoPE) in GPT-2 models trained with 512 context size on 26B token form openwebtext dataset

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

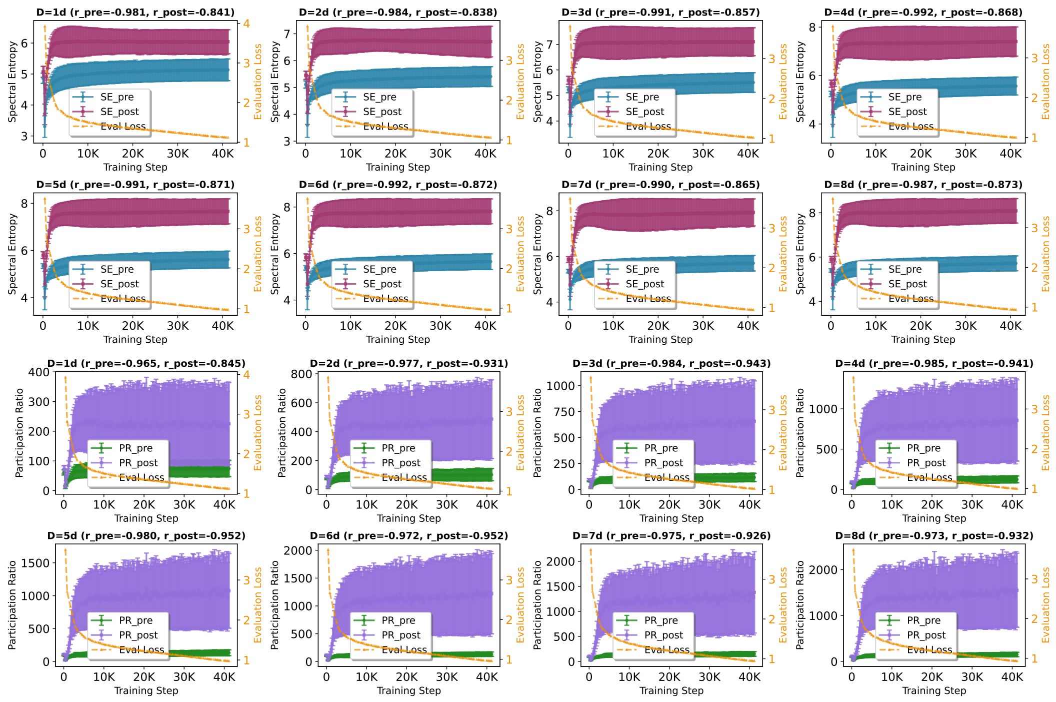

Figure 13: Eigen metrics in GPT-2 (**GELU**, D=$1d$ to $8d$) with Pearson r to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

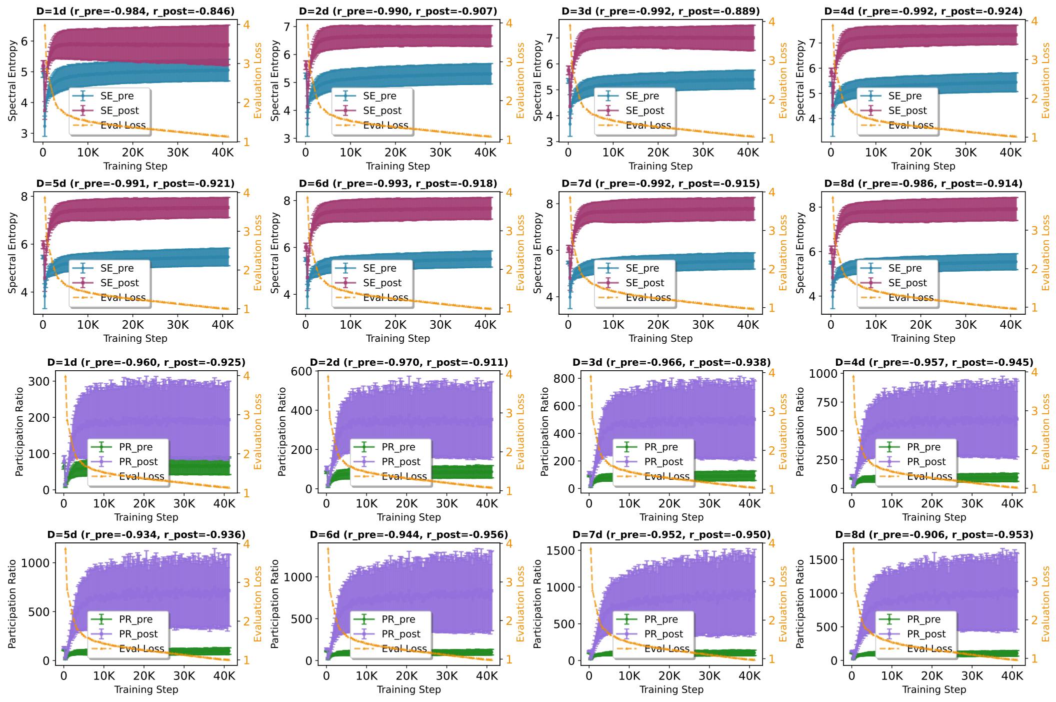

Figure 14: Eigen metrics in GPT-2 (**ReLU**, D=$1d$ to $8d$) with Pearson r to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

F Spectral Signature of Sampling and Low Rank Approximation

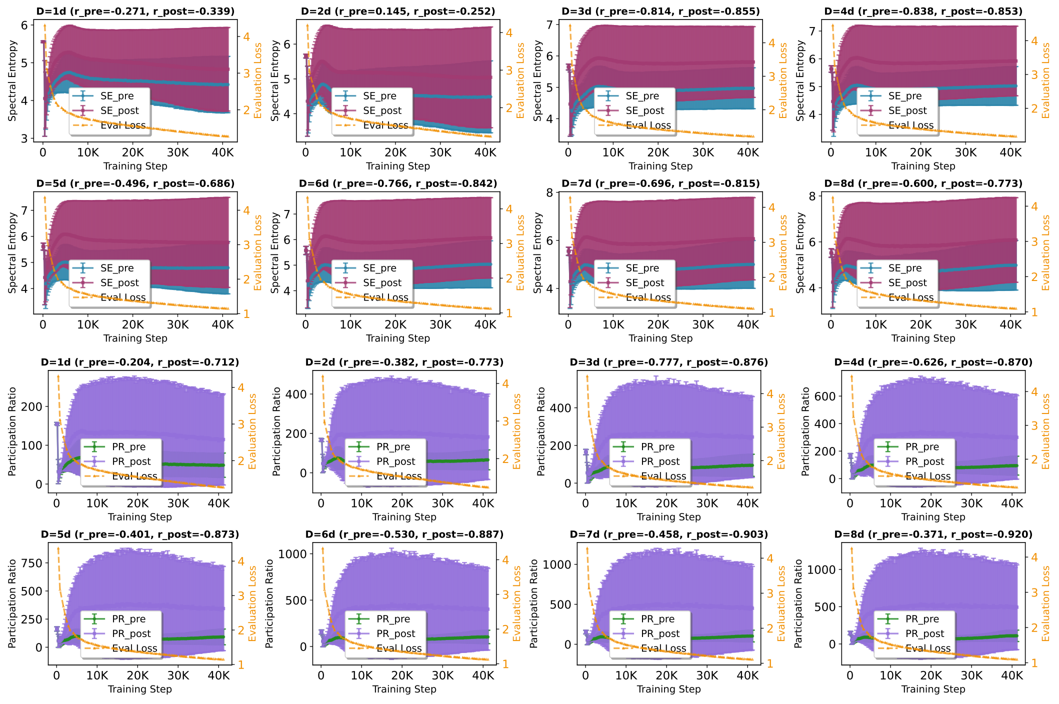

Figure 15: SE and PR in **Normalization-free** GPT-2 (**GELU**, D=$1d$ to $8d$) with Pearson correlation to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

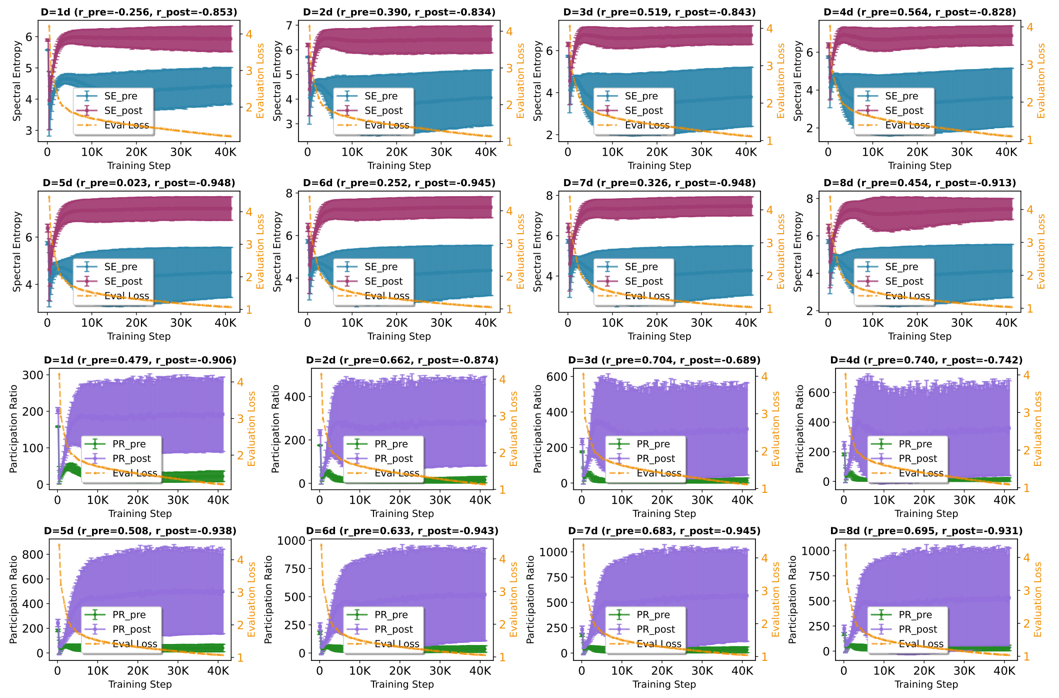

Figure 16: SE and PR in **Normalization-free** GPT-2 (**ReLU**, D=$1d$ to $8d$) with Pearson correlation to eval loss ($r_{\text{pre}}, r_{\text{post}}$)

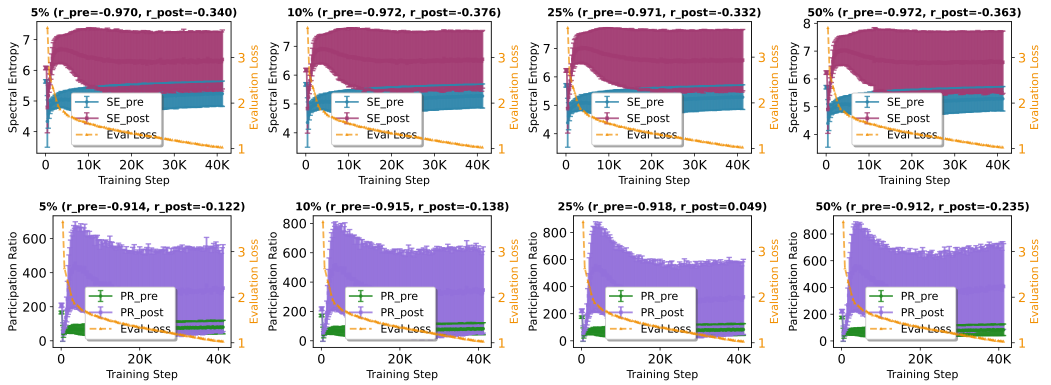

Figure 17: SE and PR in GPT-2 with Pearson r to eval loss under **sub-sampling** (5, 10, 25, 50%)

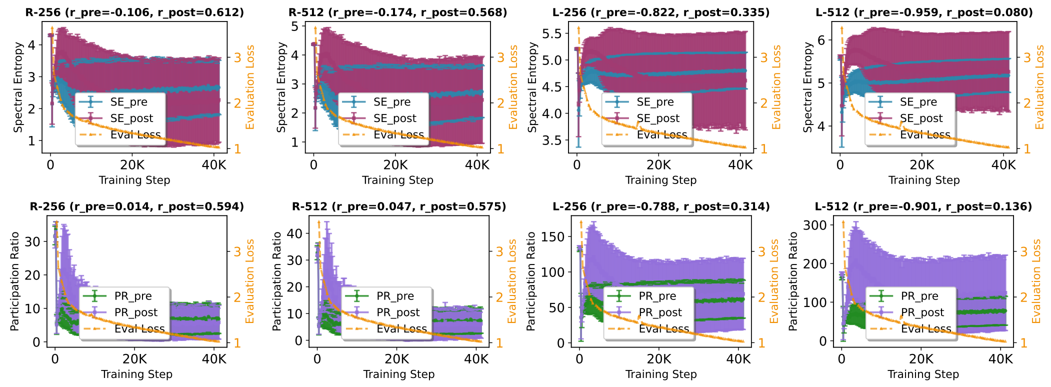

Figure 18: SE and PR in GPT-2 with Pearson r to eval loss under **low-rank approximation**.

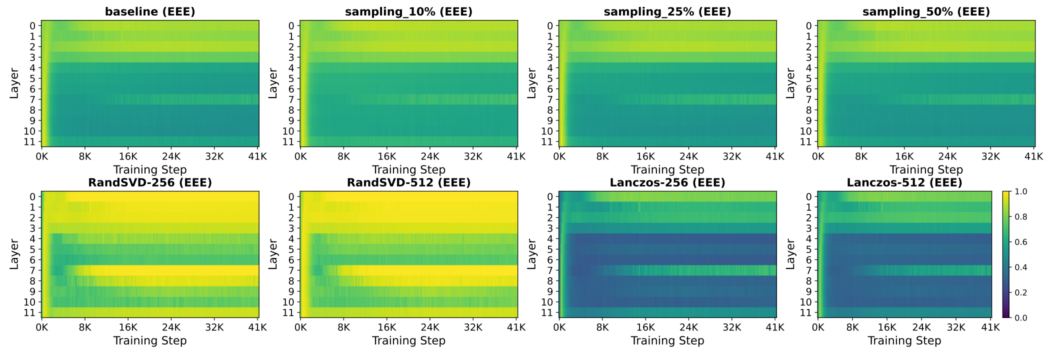

Figure 19: Impact of sampling (10%, 25%, 50%), and low-rank approximation on EEE (GPT-2)

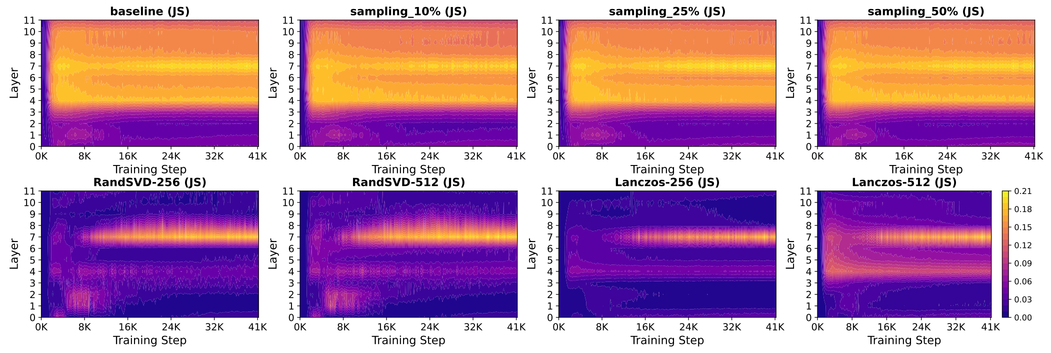

Figure 20: Impact of sampling (10%, 25%, 50%), and low-rank approximation on JS metric (GPT-2)

G Memory-efficient Eigenspectrum Analysis

```python
# 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

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.

```python
### 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:

```python
# 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 $\sim 2\times 36$MB per layer rather than accumulating $2\times 12\times 36$MB = 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.

Algorithm 1 Memory-Efficient Eigenspectrum Analysis

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

H EIGENSPECTRUM DYNAMICS IN A NON-TRANSFORMER ARCHITECTURE

We selected MLP-Mixer (Tolstikhin et al., 2021) as a non-transformer architecture family model for three key reasons: 1) it has the core architectural block that we studies in this work, wider MLPs/FFNs with LayerNorm and GELU activations, while completely removing the self-attention block; 2) the channel-mixing MLPs/FFNs in MLP-Mixer are functionally analogous to FFNs in Transformers, as they expand and squeeze the latent representation with an intervening nonlinearity, and stacked across the network's depth; and 3) it isolate the contribution of FFN nonlinear transformations from attention-specific dynamics like rank collapse in self-attention (Dong et al., 2021).

Moreover, as the MLP-Mixer is designed for vision tasks, it has fundamentally different inductive biases compared to decode-only LLMs. More importantly, it allows us to extend the eigenspectrum analysis from language modeling to vision tasks, verifying the generalization across modalities. This lets us ask: *do the eigenspectrum patterns we observe depend fundamentally on the attention mechanism, or are they a more general property of deep feedforward layers?*

Settings: Model, dataset, and training hyperparameters We use MLP-Mixer B/16 models adapted for CIFAR-100 (32×32 images, 4×4 patches, 64 tokens in one sequence) since it closely resembles the key architectural parameters from GPT-2 to enable direct comparison. The architecture consists of 12 Mixer blocks with embedding dimension 768. Each block contains two MLPs: token-mixing (hidden dimension 384) and channel-mixing (hidden dimension 3072), both using LayerNorm and GELU activation. This configuration results in 57.4M trainable parameters.

We train mixer models for 120 epochs using the Adam optimizer (Kingma, 2015) with a learning rate of $1e\text{-}3$ and weight decay of $5e\text{-}5$. We use a batch size of 128 and employ cosine annealing for learning rate scheduling with 5 epochs of linear warmup. For data augmentation, we apply AutoAugment and CutMix to improve generalization.

Methodology. We apply our NerVE framework to compute eigenspectrum metrics (pre- and post-activation) for all 12 channel-mixing MLPs (dimension 3072) at epochs 1, 10, 20, 40, 80, and 120. We use full-batch covariance estimation with no sampling. Hence, at each logging epoch, *we accumulate the statistics across the entire training dataset* in a single forward pass which yields covariance matrices from 3.2M samples per layer. Our GPU-optimized implementation accumulates statistics for all layers simultaneously, requiring $\sim$432MB of additional GPU memory.

For an extensive eigenspectral dynamics study, we evaluate four architectural variants by systematically varying the activation functions in the token-mixing (FFN1) and channel-mixing (FFN2) layers: (1) GELU, GELU (baseline); (2) GELU, ReLU; (3) ReLU, GELU; and (4) ReLU, ReLU. We apply the full NerVE analysis to all configurations, tracking eigenspectrum metrics across all 12 layers. Figure 21 shows the eigenspectrum dynamics for each configuration, demonstrating how activation function choice impacts spectral properties during training.

Observations. First, across all four settings, post-activation eigenspectrum consistently exhibit higher spectral entropy and participation ratio than their pre-activation counterparts, with the gap increasing rapidly in the first 20 to 40 epochs, indicating that the Mixer nonlinearities reliably expand the effective dimensionality of the FFN representations rather than merely reshuffling.

Second, swapping GELU for ReLU in the channel-mixing FFN (FFN2) has a much stronger effect than changing the token-mixing FFN (MLP1). Configurations with ReLU in FFN2 show larger post-activation SE/PR and a more significant drop in EEE, indicating a flatter, less top-heavy spectrum where variance is redistributed away from the leading eigenmodes.

Third, the GELU (in FFN1) and ReLU (FFN2) variant, 3rd row in Figure 21, results in most aggressively flattened spectra indicated by highest post-activation PR and lowest post-activation EEE values while maintaining small JS divergence across most layers, suggesting that ReLU in token-mixing FFN drives a more uniformly expanded, higher effective dimensionality latent space rather than inducing sharp layer-specific distortions.

Finally, JS divergence is concentrated mostly in early layers when feature-mixing (FFN2) as GELU nonlinearity, whereas ReLU in FFN2 has more uniform pattern for layerwise JS divergence, mostly concentrated in deeper layers. This suggest that activation choice mainly modulates how much the boundary layers reshape the eigenspectrum while the interior layers act as relatively stationary propagators of the learned latent representation.

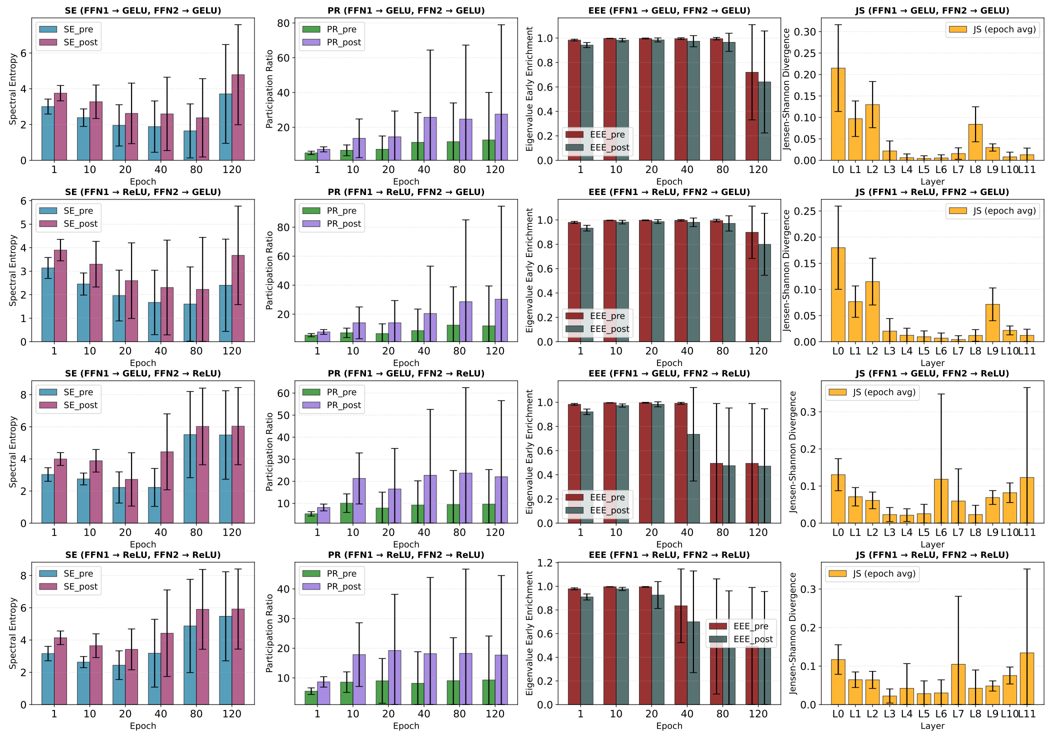

Figure 21: **Eigenspectrum dynamics in MLP-Mixer under activation ablations.** Rows correspond to the four activation configurations for token-mixing (FFN1) and channel-mixing (FFN2) layers, and columns (from left to right) show spectral entropy (SE), participation ratio (PR), eigenvalue early enrichment (EEE), and Jensen-Shannon divergence (JS) for the channel-mixing FFNs (FFN2). Each panel traces pre- and post-activation metrics over training, showing that ReLU in the channel-mixing MLP (3rd and 4th rows) most strongly increases SE/PR and reduces EEE, i.e., reinjects variance into low-energy directions and flattens the spectrum.

Accuracy on CIFAR-100 Table 7 shows the accuracy for activation ablation study in MLP-Mixer. The ReLU in FFN2 leads to a better accuracy since it flattens the spectrum (lower EEE_post) and reduce the top-heaviness of the spectrum. Whereas, GELU in FFN2 leads to a higher EEE values throughout the training. The combination of GELU in FFN1, and ReLU in FFN2 leads to highest PR and lowest EEE, indicating best utilization of latent space in FFN2 and results in best accuracy.

Table 7: Validation accuracy on CIFAR-100 for different activation function configurations in MLP-Mixer. FFN1 refers to token-mixing MLP, FFN2 refers to channel-mixing MLP.

FFN1	FFN2	Acc. (%)
GELU	GELU	66.96
GELU	ReLU	66.99
ReLU	GELU	67.86
ReLU	ReLU	67.20

I Token-Position Effects on FFN Eigenspectrum

To make the FFN eigenspectrum analysis explicitly sensitive to sequential structure, we stratify tokens by their position and recompute NerVE metrics within each position groups during their training. For GPT-2, we partition tokens into three groups along the sequence dimension (early, middle, late). For MLP-Mixer on CIFAR-100 we instead split patches into top vs. bottom rows of the 4×4 patch grid. For each configuration we then track effective dimensionality of the pre- and post-eigenspectrum using participation ratio metric, both aggregated across layers and averaged per layer (see Figure 22).

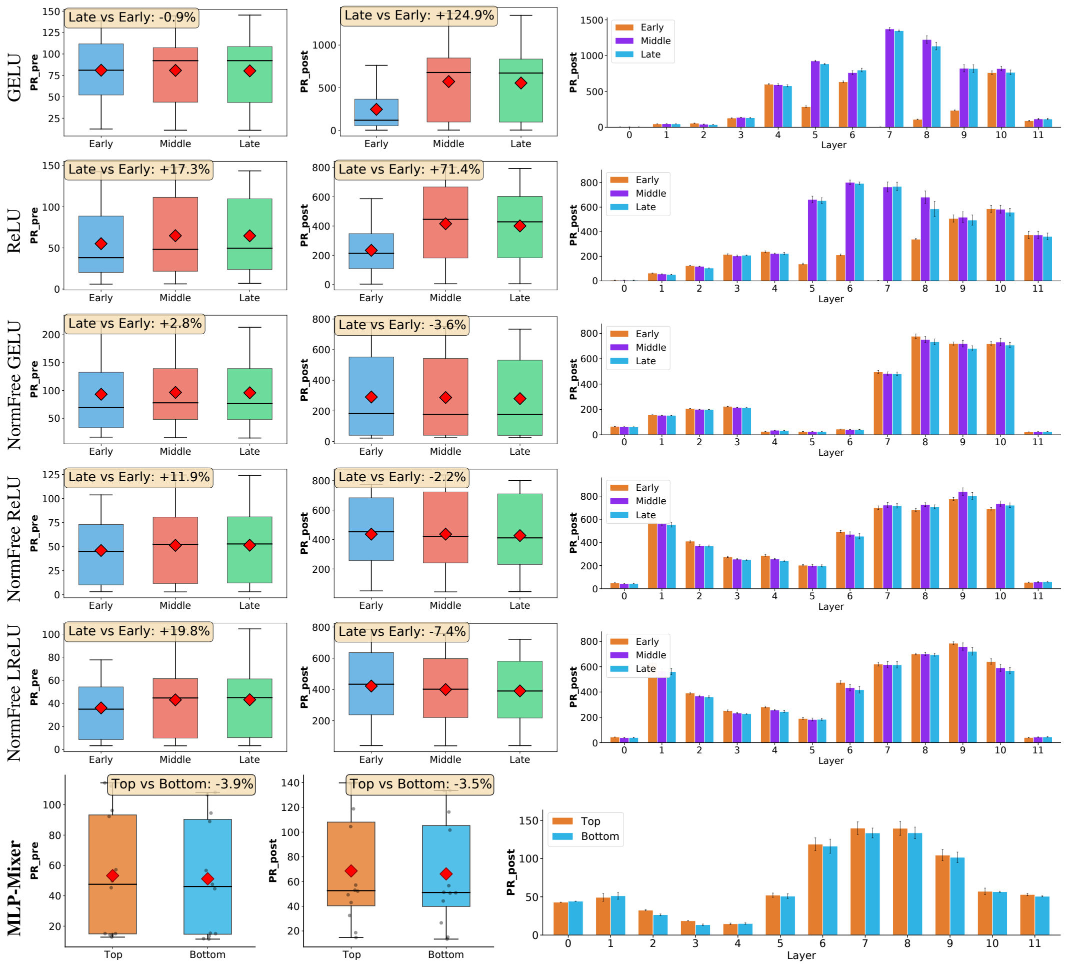

Figure 22: **Sequence-aware FFN eigenspectrum across token positions.** For four GPT-2 (125M) variants (rows 1 to 4) and a non-transformer MLP-Mixer (row 5), tokens are grouped by position. **Left**: pre-activation $\mathrm{PR}_{\mathrm{pre}}$; **Middle**: post-activation $\mathrm{PR}_{\mathrm{post}}$; and **Right**: layer-wise $\mathrm{PR}_{\mathrm{post}}$. In GPT-2 with baseline, both GELU and ReLU variants, early/middle/late tokens have similar $\mathrm{PR}_{\mathrm{pre}}$ but **diverge strongly** in $\mathrm{PR}_{\mathrm{post}}$, with late tokens using substantially higher effective dimensionality, mainly visible in deeper layers. However, in the normalization-free GPT-2 variants, these position-conditioned differences are largely suppressed in both pre- and post-activations. On the other hand, in a non-transformer family model, MLP-Mixer trained with SGD optimizer, where we split patches into top vs bottom rows, position effects are **weak** throughout. *This show that strong position-dependent FFN geometry is a characteristic of LayerNorm-based GPT-2*

Baseline GPT-2 exhibits strong position dependence only after the FFN nonlinearity. In both GPT-2 baseline variants (GELU and ReLU), the pre-activation PR shows almost no systematic differences between early, middle, and late tokens: the three boxes largely overlap, suggesting that the FFN representations before nonlinearity have comparable effective dimensionality across positions. In contrast, the post-activation PR shows a distinctive ordering: late tokens consistently exhibits a substantially higher PR than early tokens, with middle tokens in between. This effect is particularly pronounced for GELU, where late-token PR_post is roughly twice that of early tokens, and remains visible under ReLU. The layerwise trend (right column) further reveal that this distinction emerges mainly in the second half of the network: *deeper FFNs layers allocate more latent degrees of freedom to later tokens, while early layers treat positions more uniformly.*

Normalization-free GPT-2 suppresses position-dependent FFN geometry. When we remove LayerNorm layers (NormFree-GELU and NormFree-ReLU), the position-induced differences reduce dramatically. Both pre- and post-activation PR distributions for early, middle, and late tokens nearly collapse onto each other, and the layerwise trends show very small gaps between position groups at all depths. That is, once normalization is removed, the FFN eigenspectrum become almost position-agnostic. This contrast suggests that in standard GPT-2, LayerNorm coupled with the nonlinearity amplifies positional biases in the FFN latent space, which is largely disappears in the normalization-free settings.

Non-transformer MLP-Mixer shows only weak spatial position effects. For the MLP-Mixer model, we perform an analogous top-bottom split over patch positions in baseline model (GELU in both MLPs) when trained from scratch on CIFAR-100. Here, position-conditioned PR differences are small in both pre- and post-eigenspectrum, and the layerwise PR trend for top vs bottom patches almost coincide. This indicates that the strong position dependence observed in GPT-2 FFNs is *not a generic property of FFNs:* in a non-transformer architecture trained on images, NerVE sees only mild spatial variation in FFN eigenspectrum.

J FFN Eigenspectrum Dynamics: LayerNorm vs RMSNorm

J.1 LayerNorm vs RMSNorm Ablation on GPT-2

To examine how sensitive our FFN eigenspectrum analysis is to the normalization choices, we systematically vary both the placement of the normalization (PreLN, PostLN, MixLN) and the FFN activation type (GELU, ReLU, learnable leaky ReLU), and compare their effective post-activation dimensionality using the (Figure 23).

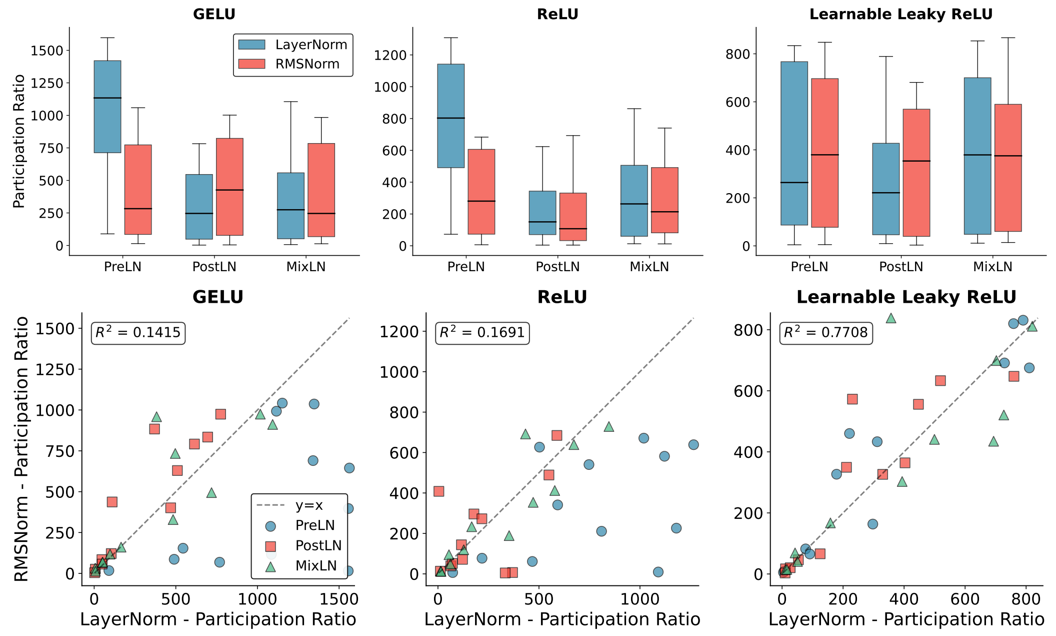

Figure 23: **Effect of LayerNorm vs RMSNorm on FFN eigenspectrum**. **Top row:** Post-activation participation ratio ($\mathrm{PR}_{\mathrm{post}}$) in GPT-2 (125M) for three FFN activations (GELU, ReLU, learnable leaky ReLU), three normalization placements (PreLN, PostLN, MixLN), and two normalization variants. Within each activation-placement pair, the LayerNorm and RMSNorm boxplots substantially overlap, while much larger shifts arise from changing the activation type or the norm placement. **Bottom row:** Layerwise comparison of $\mathrm{PR}_{\mathrm{post}}$ under LayerNorm (x-axis) vs RMSNorm (y-axis). Each point represents one FFN layer. GELU and ReLU show similar PR ranges but weak layerwise correspondence ($R^2 < 0.2$), while learnable leaky ReLU exhibits strong alignment ($R^2 \approx 0.77$). *This indicates that FFN eigenspectrum patterns are primarily driven by activation type and are largely robust to the choice of normalization scheme.*

FFN activation type and the placement of normalization layers is more consequential than LayerNorm vs RMSNorm. Figure 23 (top row) shows the distribution of post-activation participation ratio in various normalization layer positioning and FFN activation settings. Within each activation-placement pair, the LayerNorm and RMSNorm boxplots substantially overlap: their medians and interquartile ranges are of similar magnitude, and both exhibit the same depth-wise trend. For instance, PreLN exhibits highest PR spread for GELU while PostLN being more compressed.

By contrast, the *activation* choice induces much larger shifts in PR: GELU yields the largest PR values overall, ReLU compresses the spectrum, and learnable leaky ReLU lies in between with a narrower spread. Likewise, changing the PreLN to PostLN or MixLN has a clear effect on the PR distribution, whereas swapping LayerNorm to RMSNorm within a fixed placement produces only second-order changes, not the qualitative trends.

Layerwise eigenspectral structure is largely preserved for LayerNorm vs RMSNorm. Figure 23 (bottom row) compares LayerNorm and RMSNorm on a per-layer basis by plotting. For GELU and ReLU, the points cluster in a similar numeric range and follow a weak positive trend ($R^2 \approx 0.14$ and $R^2 \approx 0.17$, respectively): layers that are high-PR under LayerNorm tend to remain high-PR under RMSNorm, and low-PR layers remain low-PR, even though RMSNorm reshapes individual values somewhat. For learnable leaky ReLU, the alignment is much stronger ($R^2 \approx 0.77$). This shows that the *ordering* of layers and the qualitative depth-wise behavior of the FFN eigenspectrum are (quantitatively)robust for LayerNorm vs RMSNorm; the main axes of variation remain activation type and the placement of normalization layer.

Recall that NerVE operates on *centered* FFN activations: when we construct (pre-/post-)covariance matrices, we subtract the empirical mean across tokens. This removes the DC component for eigenspectral analysis regardless of whether the model uses LayerNorm (which performs centering inside the model) or RMSNorm (which does not). Combined with the empirical evidence above, these ablations reassure that our conclusions about FFN eigenspectrum dynamics do not rely on a particular normalization layer; they hold for both LayerNorm and RMSNorm.

J.2 LayerNorm vs RMSNorm Ablation on MLP-Mixer

We extend our LayerNorm vs RMSNorm comparison to MLP-Mixer trained on CIFAR-100. Figure 24 shows post-activation SE, PR, and EEE throughout training, and JS the final-epoch JS across layers.

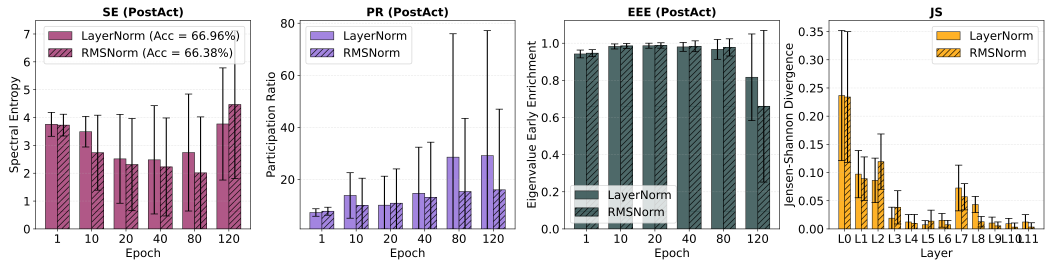

Figure 24: **Effect of LayerNorm vs RMSNorm in the MLP-Mixer models**. Columns (from left to right) shows post-activation SE, PR, EEE, and finally JS metric trend for LayerNorm and RMSNorm configurations throughout the training when MLP-Mixer models (GELU in both MLPs) are trained from scratch on CIFAR-100 dataset using Adam optimizer. FFN eigenspectrum dynamics remain (qualitatively) stable when LayerNorm is replaced with RMSNorm. However, in the later phases of training, the LayerNorm configuration attains higher effective dimensionality (PR) than RMSNorm, and its spectrum becomes noticeably flatter.

Throughout training, both normalization schemes produce very similar SE and EEE dynamics while layerwise JS trends at final convergence almost overlap, suggesting very similar reshaping of the eigenspectrum. The main difference appears in the final training phase, where the LayerNorm model attains a slightly higher PR_post and lower EEE_post than RMSNorm, and correspondingly achieves a modest accuracy gain (66.96% vs 66.38%). This suggests that, even in FFN-only architectures like MLP-Mixer, the qualitative eigenspectral behavior remain robust to the choice of normalization layer.

K OPTIMIZER-DEPENDENT FFN EIGENSPECTRUM DYNAMICS

K.1 ADAMW VS MUON VS DION

We examine the optimizer-induced eigenspectral dynamics in GPT-2 160M models, when trained from scratch on FineWeb dataset with context size 512 (Figure 26) and 1024 (Figure 25) across AdamW, Muon (Liu et al., 2025; Shah et al., 2025; Boreiko et al.) and Dion optimizers, following the architectural and training hyperparameters settings from Ahn et al. (2025)

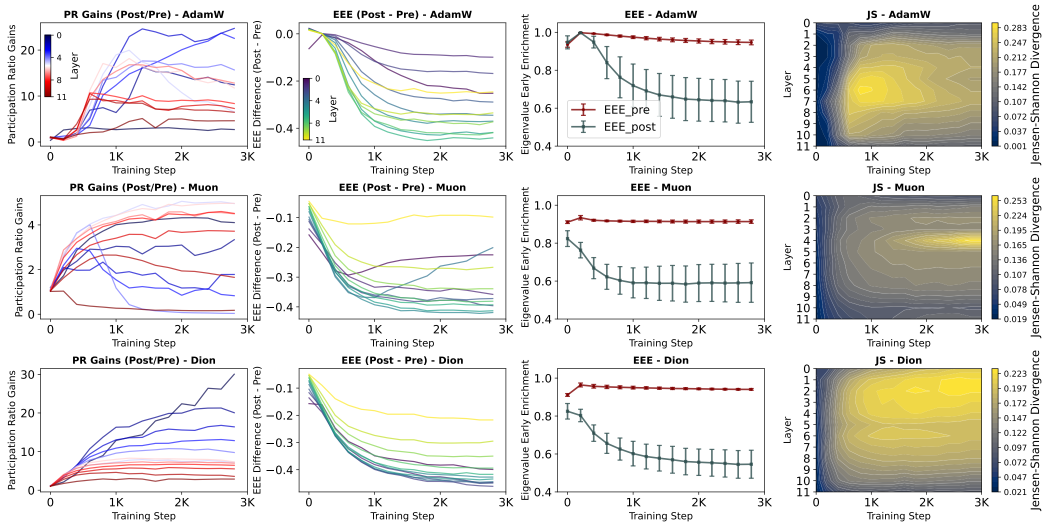

Figure 25: Optimizer-dependent FFN eigenspectrum dynamics in GPT2-160M (context length 1024). Rows show AdamW (top), Muon (middle), and Dion (bottom).

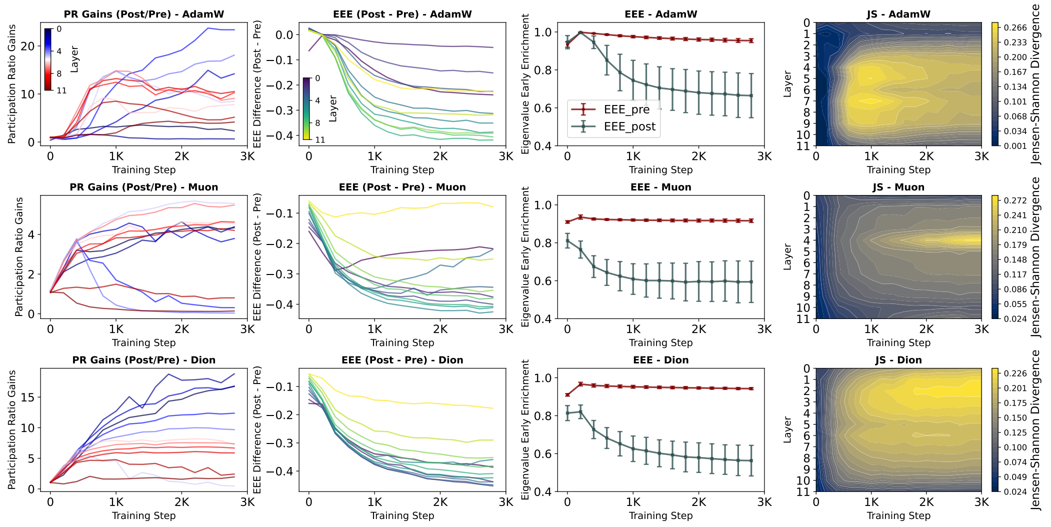

Figure 26: Optimizer-dependent FFN eigenspectrum dynamics in GPT2-160M (context length 512). Rows show AdamW (top), Muon (middle), and Dion (bottom).

Across context length 512 and 1024 in GPT-2 160M models, *a consistent eigenspectral trend emerges*. AdamW exhibits large early PR gains and high JS divergence with relatively high post-activation EEE (EEE_post), indicating optimizer-induced pre-activation collapse followed by aggressive but incomplete nonlinear repair. In contrast, Muon shows the smallest PR gains, lowest JS, and lowest EEE_post, with flatter post-activation spectra that suggest more stable eigenvalue distributions throughout training. Dion falls between these two regimes, it improves over AdamW but does not fully match Muon's spectral behavior. Notably, in a few early FFN layers, Dion exhibits PR gains on par with AdamW, suggesting layer-dependent repair dynamics.

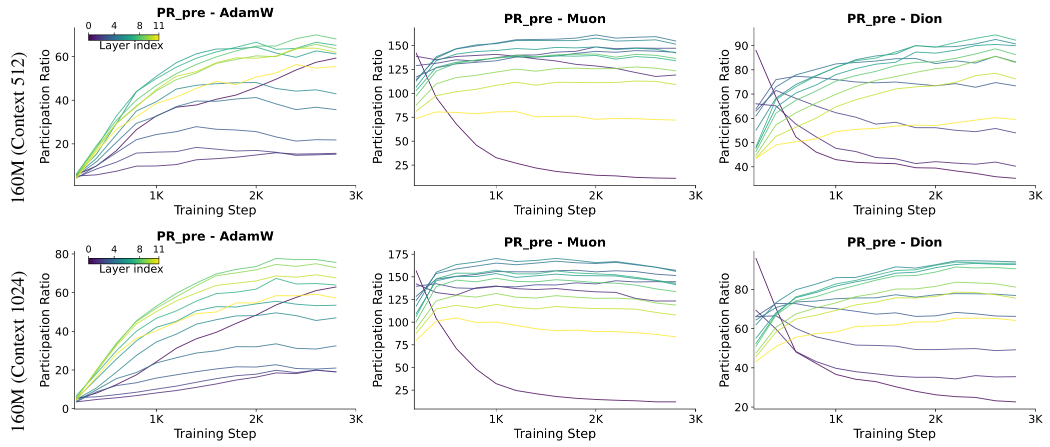

Figure 27: **Activation-compatibility of pre-activation eigenspectrum**. Columns show the effective dimensionality of pre-activation spectrum (PR_pre) when GPT2-160M models are trained from scratch with AdamW (left), Muon (middle), and Dion (right) optimizers, with 512 (top) and 1024 (bottom) context length. Muon exhibits the highest PR_pre across layers, demonstrating well-conditioned pre-activation spectra that place less burden on the FFN nonlinearity to restore representational rank.

To evaluate the conditioning of pre-activation eigenspectrum, and asses the extent to which FFN nonlinearity must actively inflate representational rank, we plot the layerwise evolution of PR_pre during training in Figure 27. Across settings, Muon consistently exhibits the highest effective dimensionality (PR_pre) across layers, indicating well-conditioned pre-activation spectra that place far less burden on the FFN nonlinearity to restore representational collapses. On contrary, AdamW shows pronounced early-layer collapse, while Dion partially mitigates these collapses.

Increasing the context length does not change this ordering; if anything, it makes the early-layer collapse under AdamW and Dion more pronounced, while Muon persistently keeps pre-spectra high-dimensional. This straighten the earlier observations that Muon produces activation-compatible representations across both model sizes and sequence lengths (Figure 8 and Figure 9).

Table 8: Evaluation perplexity (PPL) for GPT-2 models trained from scratch on the FineWeb dataset (Penedo et al., 2024) using AdamW, Muon, and Dion optimizer. Muon consistently achieves the lowest perplexity across all model sizes and context lengths, while AdamW results in worse perplexity.

Optimizer	GPT2-350M (Context 512)	GPT2-160M (Context 512)	GPT2-160M (Context 1024)
AdamW (Loshchilov & Hutter, 2019)	33.24	39.26	34.33
Muon (Jordan et al., 2024)	25.68	30.95	27.09
Dion (Ahn et al., 2025)	27.68	33.60	29.34

K.2 AdamW vs Adafactor

To further investigate the optimizer-induced effects on FFN eigenspectrum, we substitute the AdamW with Adafactor (Shazeer & Stern, 2018) in GPT-2 and evaluate both the baseline and normalization-free variants with GELU and ReLU activations. We include Adafactor for optimizer-induced ablation study because designed to reduce optimizer memory via factorized second-moment estimates for matrix-shaped parameters, and offers a different preconditioning geometry from AdamW while remaining practical for training large language models (Raffel et al., 2020).

We begin with the baseline GPT-2 (125M) with GELU and ReLU activations to substantiate the core findings about the role of nonlinearity in FFNs (see Section 3.1). Figure 28 demonstrates the similar pre- and post-eigenspectrum characteristics that we observed earlier in Figure 1 and Figure 3. Precisely, the pre-activation spectrum entering the FFN are top-heavy and anisotropic, while the post-activation spectrum exhibit higher SE and PR, and a lower EEE, indicating that the FFNs systematically reinject variance and flatten the spectrum, counteracting the rank collapse induced by self-attention (Dong et al., 2021).

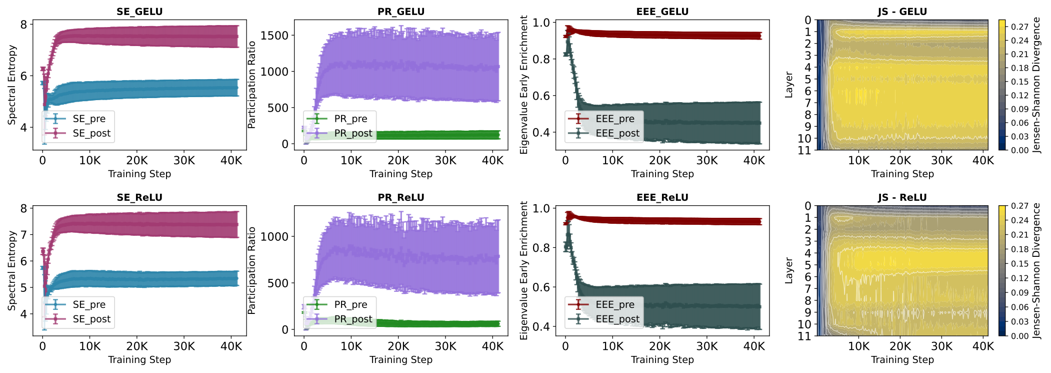

Figure 28: Eigen-metrics (SE, PR, EEE, and JS) illustrate *how FFN nonlinearities regulate information flow and reshape the eigenspectrum* when GPT-2 (125M) models with GELU (top) and ReLU (bottom) are trained from scratch on CodeParrot (2.1B tokens) datasets using **Adafactor** optimizer (Shazeer & Stern, 2018). 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 layerwise distributional shift induced by nonlinearity.

After establishing the fundamental role of FFN nonlinearity under both AdamW and Adafactor optimizer, we next compare the layerwise distribution of FFN capacity and latent-space utilization (PR_post) side-by-side in Figure 29 across four GPT-2 configurations (PreLN and Normalization-free, with GELU and ReLU activation). Notably, Adafactor consistently achieves higher post-activation PR than AdamW, and this gap is more pronounced in normalization-free ReLU model.

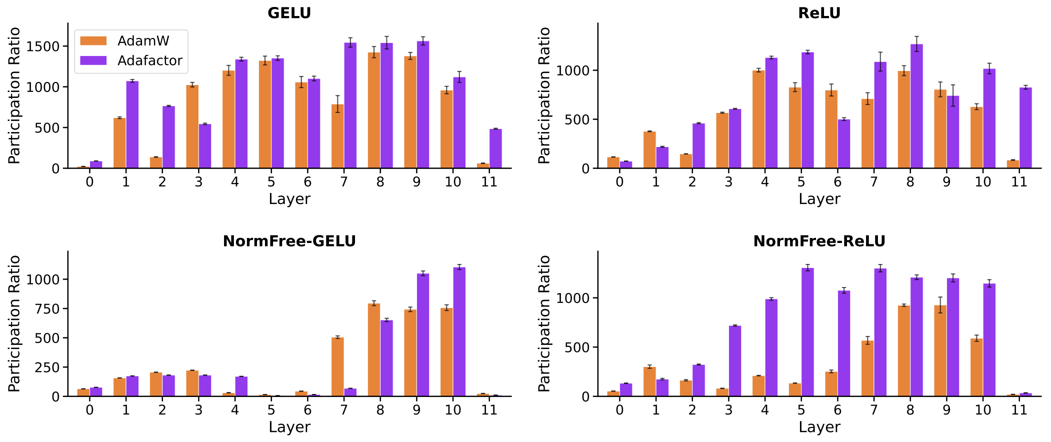

Figure 29: Layerwise post-activation participation ratio (PR_post) comparison for AdamW and Adafactor across four GPT-2 (125M) configurations: baseline (PreLN) with GELU and ReLU (top), and normalization-free GELU and ReLU (bottom). Across settings, Adafactor systematically produces higher PR_post than AdamW, indicating higher FFN latent-space utilization, with the largest gains in the normalization-free ReLU model.

This shows that, while FFN nonlinearity plays the same qualitative role—expands and flattens the spectrum—under both AdamW and Adafactor, the degree of this expansion is optimizer-dependent. Adafactor consistently drives stronger spectral expansion, activating more FFN latent capacity, especially in the normalization-free ReLU model.

Further, to contrast how effective dimensionality evolves under these two optimizers (AdamW and Adafactor), we plot the layerwise post-activation participation ratio (PR_post) over the entire training run (Figure 30). Across all four configurations, deeper layers quickly rise to a substantially higher PR value, and maintain that higher PR_post throughout the training. *This effect is consistently stronger with Adafactor than with AdamW*, indicating that Adafactor activates more latent capacity in the later FFNs throughout training.

We also track the change in EEE, $\Delta\text{EEE} = \text{EEE}_{\text{post}} - \text{EEE}_{\text{pre}}$, over training (Figure 31). More negative ΔEEE indicates stronger suppression of top eigenvalues and a flatter spectrum. Across all four

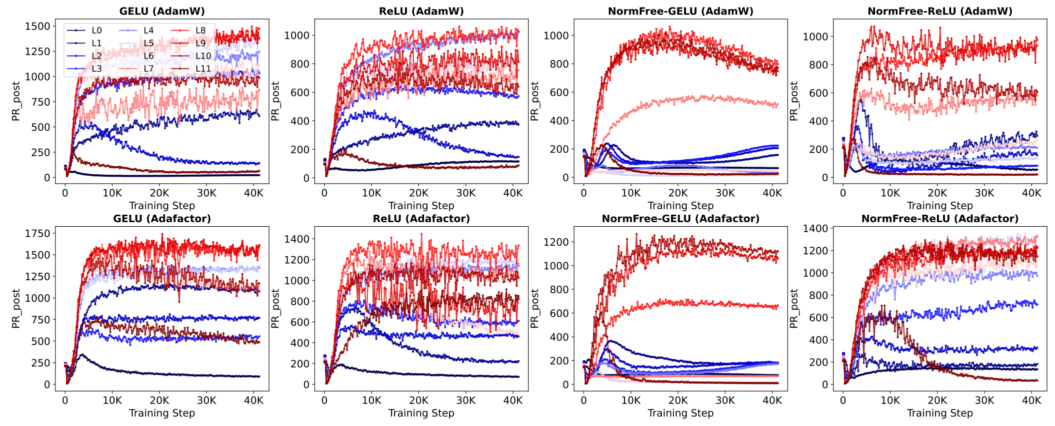

Figure 30: Training dynamics of post-activation participation ratio (PR) for all layers in GPT-2 (125M) across four configurations (columns) and two optimizers (rows). Deeper layers achieve and maintain higher PR_post throughout training, with Adafactor consistently producing larger PR_post than AdamW, especially in the deeper FFNs and in the normalization-free ReLU models.

GPT-2 configurations, both AdamW and Adafactor reduces ΔEEE well below zero, confirming the FFN's spectral flattening role. However, in the deeper layers, Adafactor consistently attains more negative ΔEEE than AdamW, especially in the normalization-free ReLU model, indicating a stronger reduction of top-heaviness and more aggressive spectral flattening in those FFNs.

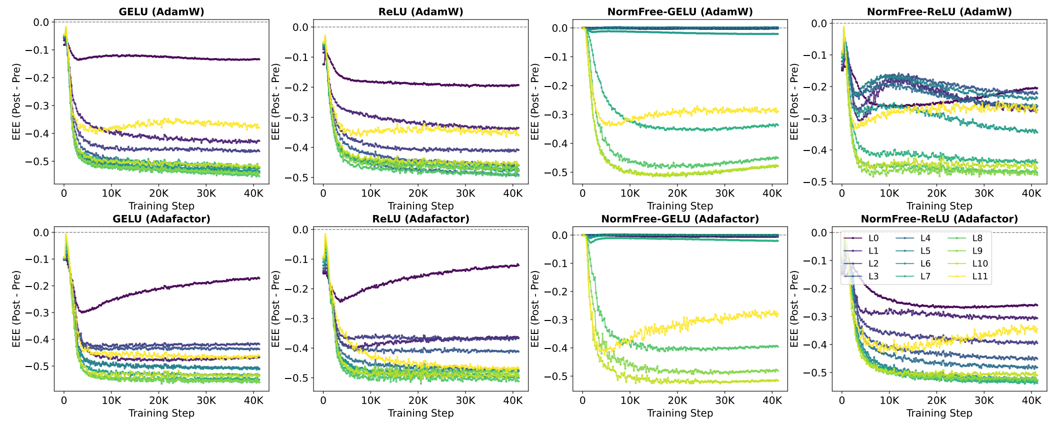

Figure 31: Training dynamics of FFN-induced spectral flattening in GPT-2 (125M), quantified as $\Delta\text{EEE} = \text{EEE}_{\text{post}}$ - EEE_{pre}, under AdamW (top row) and Adafactor (bottom row) optimizer. More negative ΔEEE indicates stronger suppression of top eigenvalues and a flatter spectrum. Across settings, deeper layers under Adafactor tend to reach more negative ΔEEE than under AdamW, indicating more aggressive spectral flattening.

K.3 AdamW vs SGD

To examine how optimizers shape latent space utilization in a non-Transformer setting, we train the MLP-Mixer baseline (GELU in both MLPs) on CIFAR-100 with SGD and compare its eigenspectrum characteristics side-by-side with the Adam variant in Figure 32. For SGD, we use a learning rate of $5e$-2, momentum 0.9, and weight decay $1e$-4.

SGD outperforms Adam on MLP-Mixer (68.07% vs. 66.96%), and their spectral metrics trend explains this performance gap. Notably, SGD attains a significantly higher (post-activation) spectral entropy and participation ratio throughout training, indicating higher effective dimensionality and better utilization of FFN representational capacity. The EEE trend shows that Adam remains near 1.0 throughout training which suggests the persistent concentration of variance in the top eigenvalues. JS divergence further shows that Adam induces stronger spectral transformation in the very first layer, whereas SGD-induced transformation between pre- and post-eigenspectrum remain lower and more uniform across depth.

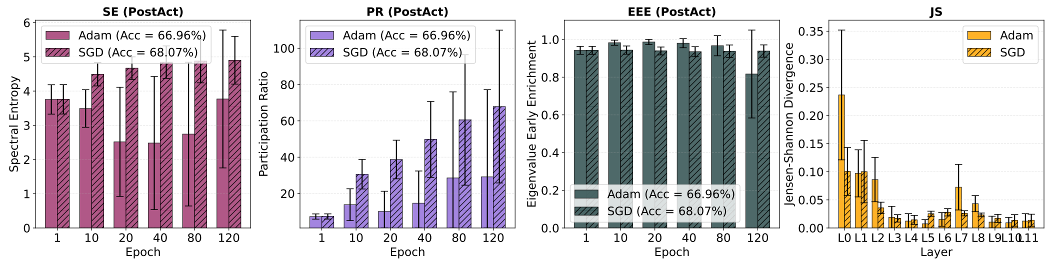

Figure 32: **Optimizer effects in MLP-Mixer (Adam vs. SGD)**. Columns (from left to right) shows post-activation SE, PR, EEE, and finally JS divergence metric trend when MLP-Mixer baseline model (GELU in both MLPs) is trained from scratch on CIFAR-100 dataset using Adam and SGD optimizer. Right from the epoch 10, the MLP-Mixer with the SGD optimizer starts exhibiting superior SE_post and PR_post showing significantly better utilization of FFN2 latent space.