Impact of Layer Norm on Memorization and Generalization in Transformers

Rishi Singhal
Department of Computer Science
North Carolina State University

Jung-Eun Kim*
Department of Computer Science
North Carolina State University

Abstract

Layer Normalization (LayerNorm) is one of the fundamental components in transformers that stabilizes training and improves optimization. In recent times, Pre-LayerNorm transformers have become the preferred choice over Post-LayerNorm transformers due to their stable gradient flow. However, the impact of LayerNorm on learning and memorization across these architectures remains unclear. In this work, we investigate how LayerNorm influences memorization and learning for Pre- and Post-LayerNorm transformers. We identify that LayerNorm serves as a key factor for stable learning in Pre-LayerNorm transformers, while in Post-LayerNorm transformers, it impacts memorization. Our analysis reveals that eliminating LayerNorm parameters in Pre-LayerNorm models exacerbates memorization and destabilizes learning, while in Post-LayerNorm models, it effectively mitigates memorization by restoring genuine labels. We further precisely identify that early layers LayerNorm are the most critical over middle/later layers and their influence varies across Pre and Post LayerNorm models. We have validated it through 13 models across 6 Vision and Language datasets. These insights shed new light on the role of LayerNorm in shaping memorization and learning in transformers[2].

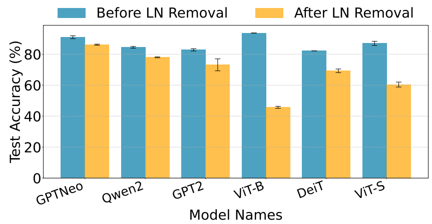
1 Introduction

Layer Normalization [Lei Ba et al., 2016] greatly contributes to stabilizing training and optimizing performance in deep learning models, especially in transformers. It works by normalizing the activations at each layer, ensuring a more consistent gradient flow during training. In recent years, transformers are primarily designed with two options by LayerNorm (LN) placements: *Pre-LN* and *Post-LN*. Post-LN transformer, introduced by Vaswani et al. [2017], applies normalization after the addition of the layer's output with the residual connection's output and has been showing competent performance in language modeling and machine translation. However, due to the issue of unstable gradient flow [Liu et al., 2020] in Post-LN models, Pre-LN transformers [Xiong et al., 2020] were introduced, where normalization is applied before self-attention and feed-forward layers. This configuration stabilized training and achieved faster convergence by improving the gradient flow, making it the preferred choice in modern architectures such as GPT, Llama, and Vision Transformers.

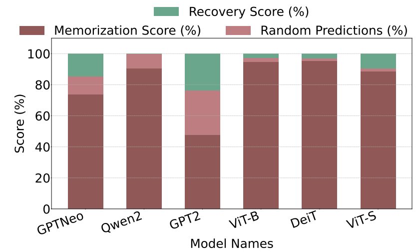
Even though transformers have demonstrated remarkable capabilities in learning rich representations from data, they exhibit a strong tendency to memorize some samples due to their complex nature, which is commonly known as *Label Memorization* [Feldman, 2020, Feldman and Zhang, 2020], where a model memorizes labels in training without learning the relevant patterns that generalize to unseen data, leading to overfitting. Recent studies have explored whether memorization can be localized to specific layers [Maini et al., 2023, Baldock et al., 2021] or components such as attention heads and the feed-forward network (FFN) [Haviv et al., 2023, Geva et al., 2023, Yu et al., 2023]. Apart from the attention heads and the feed-forward network (FFN), LayerNorm stands as a

*Corresponding author
[2]Code available at: `https://github.com/JEKimLab/NeurIPS2025_LayernormMemorization`

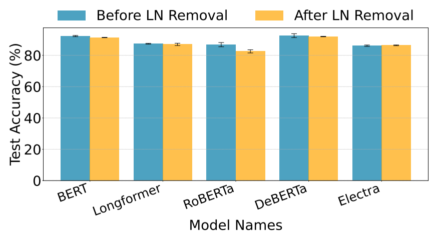
39th Conference on Neural Information Processing Systems (NeurIPS 2025).



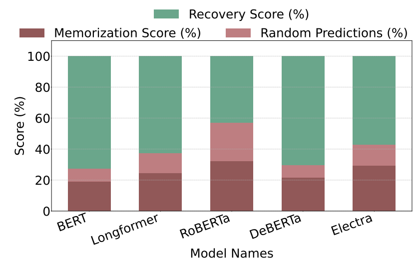
(a) Learning (test) accuracy for Pre-LN models



(b) Memorization, Recovery & Random prediction scores for Pre-LN models



(c) Learning (test) accuracy for Post-LN models



(d) Memorization, Recovery & Random prediction scores for Post-LN models

Figure 1: **Impact of LN layer on memorization and learning of Pre- and Post-LN models.** (a) shows a clear impact of LN in Pre-LN models, whereas (c) shows no impact of the removal of LN parameters in Post-LN models for learning. (b) exhibits that, without LN layers, the Pre-LN models struggle with high memorization and random predictions (red-color-family bars), while (d) exhibits that in Post-LN models, removing LN parameters recovers a significant portion of correct predictions (green bars).

pivotal component in the transformer architecture that further shapes its optimization dynamics and performance. Outlier neurons in LN layers have been shown to impair transformer performance and hinder quantization [Kovaleva et al., 2021, Puccetti et al., 2022, Bondarenko et al., 2023, He et al., 2024]. [Xu et al., 2019] hints that LN may contribute to increased overfitting in Pre-LN models.

In our paper, we identify that in Pre-LN transformers, LN is critical to *learning* and removal of its learnable parameters exacerbates overfitting and disrupts learning. On the contrary, in Post-LN transformers, LN plays a significant role in *memorization*, where by eliminating LN parameters, memorization is suppressed by recovering the true genuine labels without affecting the model's learning capability. We rigorously validate our claims through various models - BERT, Longformer, RoBERTa, DeBERTa, ELECTRA, DistilBERT, GPTNeo, GPT2, Qwen2, RoBERTa-PreLayerNorm, ViT-Base, ViT-Small, and DeiT, across both Vision and Language tasks. In summary, the core findings of our paper regarding the impact of LN on memorization and learning in transformers are as follows:

- **Learning Stability, Memorization Suppression & Label Recovery:** We identify that LN is crucial for learning in Pre-LN models, unlike Post-LN models. For Post-LN models, LN learnable parameters removal suppresses memorization and recovers genuine labels, whereas in Pre-LN models, LN removal exacerbates overfitting, with persistent memorization.

- **Early LNs are Critical:** We uncover that removal of LNs parameters in early layers is most impactful in mitigating memorization for Post-LN models, and destabilizing the learning in Pre-LN architectures.

- **Gradients Explain LN's Impact:** We explain the divergent impacts of LN in Pre- and Post-LN models by comparing learning and memorization gradients, which reveal why LN parameter removal causes learning disruption and memorization suppression in Pre- and Post-LN models, respectively.

2 Related Works

Memorization & Learning: Transformers excel at learning general, simple patterns [Arpit et al., 2017, Shah et al., 2020, Zhou and Wu, 2023], but also tend to memorize rare, mislabeled, or complex examples [Stephenson et al., 2021, Baldock et al., 2021, Agarwal et al., 2022]. Feldman and Zhang [2020], Feldman [2020] formally define label memorization, while Baldock et al. [2021] proposes prediction depth to capture example difficulty. Other works [Jiang et al., 2020, Ravikumar et al., 2024, Garg et al., 2023] associate high curvature and consistency with long-tailed or mislabeled samples. Beyond identifying memorization, several studies [Haviv et al., 2023, Geva et al., 2023, Dai et al., 2021] investigate how self-attention and feedforward layers contribute to factual recall across transformer layers. More recent work [Yin et al., 2023, Lad et al., 2024, Men et al., 2024, Li et al., 2024, Sun et al., 2025] highlights the limited effectiveness of deeper layers on learning in Pre-LN transformers. Despite these insights, the distinctive impact of LayerNorm in shaping memorization and learning across both Pre- and Post-LN architectures remains poorly understood.

Understanding LayerNorm (LN) in Transformers: In addition to self-attention and feedforward networks (FFNs), Layer Normalization (LN) plays a critical role in transformer models. Prior work [Brody et al., 2023, Wu et al., 2024] has demonstrated that LN is essential to the overall expressivity of transformers. Beyond its utility, LN has been found to contain outlier neurons [Kovaleva et al., 2021, Puccetti et al., 2022], whose removal severely degrades model performance. These outliers have also been shown to hinder the quantization of transformer models [Bondarenko et al., 2023, He et al., 2024]. Moreover, several studies [Xiong et al., 2020, Liu et al., 2020, Takase et al., 2022, Xie et al., 2023, Kim et al., 2025] have highlighted that Post-LN architectures can cause gradient instability during training, while Pre-LN configurations may lead to exploding gradients in early layers—prompting the development of techniques [Shleifer et al., 2021, Wang et al., 2022, Kumar et al., 2023, Qi et al., 2023, Jiang et al., 2023] to address them. Additionally, Xu et al. [2019] suggested that LN parameters may contribute to overfitting in Pre-LN models.

However, we provide a far more nuanced understanding of LN's role: in Pre-LN transformers, LN is essential for learning but not memorization, whereas in Post-LN models, LN is crucial for memorization but not learning. This distinction offers a novel contribution to understanding the function of LN in transformers for learning and memorization.

3 Prelimnaries

3.1 Understanding LayerNorm in Transformers and Defining Memorization & Learning

LayerNorm Operation. Let $x = (x_1, x_2, \ldots, x_d)$ be the input of size d to the LayerNorm function $\text{LN}(x)$ which first normalizes the input x as $N(x)$ using mean μ and standard deviation σ. Then it re-scales and re-centers $N(x)$ using the learnable parameters w (weight) and b (bias). The output of the LayerNorm layer is then given by:

$$\text{LN}(x) = w \odot N(x) + b, \quad \mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \quad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}, \quad N(x) = \frac{x - \mu}{\sigma}, \tag{1}$$

where $\odot$ denotes the dot product operation.

Pre-LN & Post-LN Transformers. In the Pre-LN Transformer, LayerNorm is applied before each sub-layer - Multi-Head Self Attention (MHSA) and Feed-Forward Network (FFN). On the other hand, in the Post-LN Transformer, LN is applied after the residual connection. We represent the key difference in the architectural design of the two configurations as follows:

$$\begin{aligned} &\textbf{Pre-LN:} \quad x' = x + \text{MHSA}(\text{LN}_1(x)) \qquad &&\textbf{Post-LN:} \quad x' = \text{LN}_1(x + \text{MHSA}(x)) \\ &\qquad\qquad\; y = x' + \text{FFN}(\text{LN}_2(x')) &&\qquad\qquad\;\; y = \text{LN}_2(x' + \text{FFN}(x')) \end{aligned} \tag{2}$$

Understanding Learning and Label Memorization (LM). Deep neural network models, like transformers, learn meaningful relationships between features and labels during training and generalize the learned representations to unseen test data - the phenomenon is well understood as ***learning/generalization***. At the same time, these models also have the tendency to memorize training data points which are complex in nature, commonly known as ***label memorization (LM)*** [Feldman, 2020, Feldman and Zhang, 2020], where the model just memorizes the labels during training without capturing meaningful patterns that generalize to new data, resulting in overfitting. Label memorization is known to occur due to multiple factors such as complex, ambiguous features, and noisy labels [Baldock et al., 2021], which makes it difficult for the model to learn any meaningful relationship.

In this work, we specifically focus on introducing *noisy labels* as a way to study memorization, where we change the label of a particular class sample to a randomly chosen class label that is different from its original label. To ensure that the noisy label samples are memorized, we train the model until it achieves 100% training accuracy. Throughout our experiments, we introduce random label noise in all datasets by modifying 1% of the training set labels, maintaining consistency across evaluations.

3.2 Investigating LayerNorm (LN) Impact on Memorization and Learning

Removing LN parameters. To examine the role of LayerNorm (LN) in memorization and learning within transformers, we analyze the effects of omitting its learnable parameters, during training. This provides insights into how LN influences the balance between learning and memorization for both Pre- and Post-LN models. We precisely analyze the impact of LN on memorization and learning in Sec. 4. Please note that we use *LN removal* and *LN parameters removal* interchangeably in our paper. They both refer to removal of learnable parameters of the LN layer, while keeping the normalization operation, $N(x)$, intact.

Effect of LN Removal across Layers. To further understand the impact of LN at different stages of the model, we categorize the layers into - early, middle, and later layers (described in detail in Appendix F.4). We then selectively remove LN parameters from one set at a time to analyze how their absence affects learning and memorization. This analysis reveals which set of LNs is the most influential towards memorization and learning behaviors in Pre- and Post-LN transformers. The experiments and results are discussed in Sec. 5.

LN Gradients Analysis across Layers. To support our observations on the influence of LN, we compute the gradient of the loss function ($\mathcal{L}$) with respect to the input of LN, x, represented as $\nabla_x \mathcal{L}$ or $g_x = \frac{\partial \mathcal{L}}{\partial x}$. This measure quantifies how much the input to the LN layer affects the model's loss, thereby its learning and memorization ability. To understand the sensitivity of each layer's LN towards memorization and learning, we compute the L2-norm of this gradient (i.e., $\|g_x\|_2$). Specifically, to quantify sensitivity towards *learning*, i.e., how the model generalizes the patterns to the test set, we compute $\|g_x\|_2$ for every test-set sample and average it across all of them, obtaining ***learning gradient norm***, denoted by $\|g_x^{\text{learn}}\|_2$. For *memorization*, we compute $||g_x||_2$ for each of the *noisy labels* samples that we injected into the train set and then averaged across all such noisy samples to obtain ***memorization gradient norm***, denoted by $\|g_x^{\text{mem}}\|_2$.

A higher gradient norm indicates that the layer's LN significantly influences the model's ability to memorize or learn, while a lower gradient suggests minimal impact. The discussion of memorization and learning gradients and their significance is shown in Sec. 6.

3.3 Key Metrics: Learning Accuracy, Memorization, Recovery & Random Predictions Score

To evaluate the impact of LN on the learning and memorization ability of the transformer models, we focus on several key metrics that provide insights into their behavior and effectiveness in the presence of noisy labels during training.

Learning (Test) Accuracy (%) refers to the model's performance on the test set, depicting how well it generalizes the learnt relationships to unseen data, marking it as a core indicator of the model's learning progress ($\frac{\text{\#Correct predictions on test set}}{\text{\#Total test set samples}} \times 100$). A high learning accuracy signifies that the model has learned meaningful patterns and is able to generalize well to unseen data. On the contrary, a low learning accuracy depicts poor generalizability.

Memorization Score (%) serves as an indicator of the model's tendency to memorize noisy labels that are irrelevant or erroneous, rather than genuinely learning the true underlying relationships ($\frac{\text{\#Noisy label samples memorized}}{\text{\#Total noisy label samples}} \times 100$). A high memorization score indicates that the model has overfit the noisy labels, effectively "memorizing" them.

Recovery Score (%) is a crucial metric that helps in understanding the impact of LayerNorm (LN) on memorization of noisy labels. It measures the model's ability to recover the genuine, true labels after the removal of LN parameters ($\frac{\text{\#Recovered noisy label samples as true labels}}{\text{\#Total noisy label samples}} \times 100$). A high recovery

Table 1: Summary of the impact of LN layer in Pre- and Post-LN Models.

If LN Removed	Learning Intact?	Memorization Mitigated?	Recovery Happens?
Pre-LN Model	✗ *Learning Disrupted*	✗ *Memorization Still Present*	✗ *Negligible Recovery*
Post-LN Model	✓ *Stable Learning*	✓ *Memorization Mitigated*	✓ *Genuine Labels Inferred*

score indicates that the model can successfully recover the original, correct labels by suppressing memorization.

Random Prediction Score (%) measures the percentage of noisy label samples whose predictions were changed to random labels after the removal of LN parameters. These predicted random labels are neither genuine nor the noisy label ($\frac{\text{\#Random predictions of noisy label samples}}{\text{\#Total noisy label samples}} \times 100$). Although this is not ideal, it provides a complete picture of the impact of LN parameters removal and indicates the extent to which the model can recover the true labels. A high percentage of random predictions suggests that the model struggles to recover the true labels effectively.

3.4 Datasets & Models Used

We empirically verify all claims and show extensive results against both language and vision modalities, including 3 language and 3 vision classification datasets, and 7 Pre-LN and 6 Post-LN transformers architectures, as follows:
Datasets: CIFAR10 [Krizhevsky et al., 2009], NICO++ [Zhang et al., 2023], UTK-Face [Zhang et al., 2017], Emotions [Saravia et al., 2018], News [Okite97, 2024], and TweetTopic [Antypas et al., 2022]
Post-LN Models: BERT [Devlin et al., 2019], RoBERTa [Yinhan et al., 2019], DistilBERT [Sanh et al., 2019], DeBERTa [He et al., 2020], ELECTRA [Clark, 2020], and Longformer [Beltagy et al., 2020]
Pre-LN Models: ViT-B [Alexey, 2020], ViT-S [Assran et al., 2022], DeiT [Touvron et al., 2021], GPT2 [Radford et al., 2019], GPT-Neo [Black et al., 2022], Qwen2 [Yang et al., 2024], and RoBERTa-PreLayerNorm [Ott et al., 2019].

It needs to be acknowledged that only language modality is available for the Post-LN architecture in practice/literature. We provide a thorough discussion of the datasets, models, and training configurations in Appendix F. All our experiments are run across 3 random seeds.

4 Impact of LN on Memorization and Learning

In this section, we examine the distinct impact of Layer Normalization (LN) on memorization and learning in Pre-LN and Post-LN transformers. To assess its influence, we train two versions of the model — one with LN parameters removed and one with them intact — and compare their performance using learning accuracy, memorization, recovery, and random prediction scores.

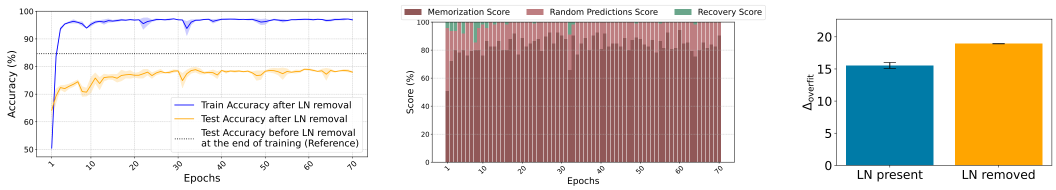
4.1 Learning Stability

From Figs. 1a & 1c, we observe that removing LN parameters in Pre-LN transformers significantly disrupts learning, whereas Post-LN transformers remain robust, maintaining their learning accuracy even after LN parameter removal.

This discrepancy becomes even more evident when analyzing the progression of learning in epochs, as depicted for Qwen2 (Pre-LN) in Fig. 2a & ELECTRA (Post-LN) in Fig. 2d. For Qwen2, once learning is disrupted by LN parameters removal, it does not recover till the end of training, indicating a fundamental instability. However, ELECTRA maintains stable learning throughout training, showing no signs of degradation, further highlighting its resilience to LN parameters removal. Similar results are observed for other Post-LN (BERT, DeBERTa, Longformer, RoBERT) and Pre-LN models (GPT2, GPTNeo, ViT-B, DeiT, ViT-S), as shown in Appendix G.1.

4.2 Memorization Suppression & Label Recovery

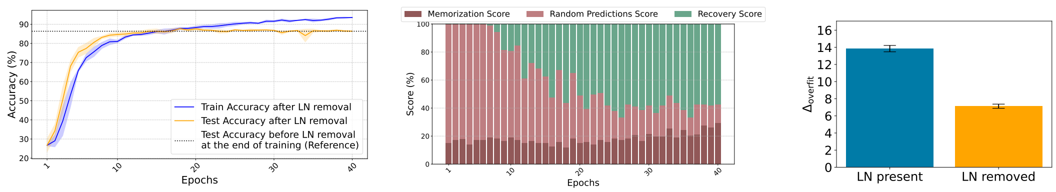
We now examine the role of LN in memorization and label recovery. From Figs. 1b & 1d, we observe that in Post-LN models, LN governs memorization, its parameter removal mitigates memorization and enhances true label recovery, reflected in lower memorization scores and higher recovery scores.



(a) Learning (Test) accuracy over epochs for Pre-LN Model (Qwen2)

(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (Qwen2)

(c) Overfitting gap for Pre-LN Model (Qwen2)



(d) Learning (Test) accuracy over epochs for Post-LN Model (ELECTRA)

(e) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (ELECTRA)

(f) Overfitting gap for Post-LN Model (ELECTRA)

Figure 2: **LN removal destabilizes learning in Pre-LN models, while mitigates memorization in Post-LN models (News Dataset):** LN removal in Pre-LN models critically affects learning (accuracy gap in (a)) while Post-LN models remain robust (negligible gap in (d)); LN removal helps in effective mitigation of memorization and high recovery in Post-LN models (green bars in (e)), while memorization/random predictions still persist in Pre-LN models (red-color-family bars in (e)); LN removal in Pre-LN models exacerbates overfitting explained by increasing train-test accuracy gap in (c), and for Post-LN models it decreases due to memorization mitigation (see (f)).

In contrast, for Pre-LN models, LN parameters removal does not mitigate memorization, as indicated by persistently high memorization and random prediction scores.

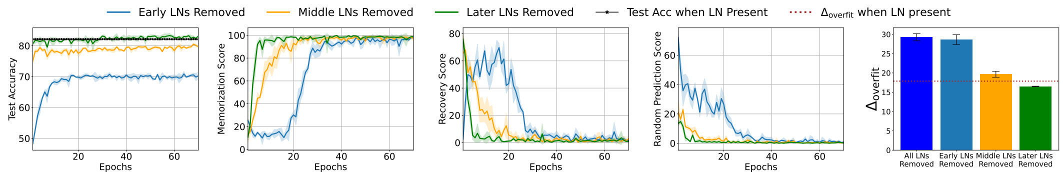
This effect is even clearer when analyzing memorization over epochs, as shown for ELECTRA (Fig. 2e) and Qwen2 (Fig. 2b). In ELECTRA, memorization decreases over time, with label recovery improving as training progresses. Conversely, in Qwen2, memorization persists throughout training, and label recovery remains poor, indicating that LN parameter removal does not suppress memorization or aid label recovery in Pre-LN models. Similar patterns are observed in other Post-LN (BERT, DeBERTa, Longformer, RoBERT) and Pre-LN models (GPT2, GPTNeo, ViT-B, DeiT, ViT-S), as shown in Appendix G.1. These findings offer a nuanced perspective on LN's role: it is crucial for learning in Pre-LN models but does not influence memorization, contrary to prior work [Xu et al., 2019], which suggested that LN in Pre-LN models can contribute to overfitting.

In summary, **LN is essential for stable learning in Pre-LN models**, hence its parameters removal significantly destabilizes learning and widens the train-test accuracy gap ($\Delta^{\text{Pre}}_{\text{overfit}}$), i.e., exacerbating overfitting/memorization as illustrated in Fig. 2c. In contrast, in **Post-LN models, LN parameters removal suppresses memorization and enhances true label recovery**, thereby narrowing the train-test accuracy gap ($\Delta^{\text{Post}}_{\text{overfit}}$) as shown in Fig. 2f. This distinction is further illustrated in Table 1, which provides a comparative overview of LN's role in learning and memorization across Pre-LN and Post-LN model. Similar observations are observed in other Pre-LN (GPT2, GPTNeo, ViT-B, ViT-S, DeiT) and Post-LN (BERT, RoBERTa, DeBERTa, Longformer) models as reported in Appendix G.1.

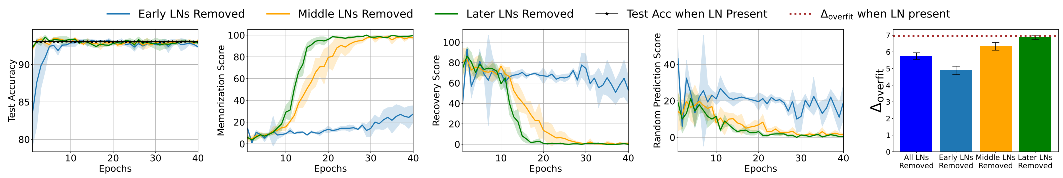
5 The Pivotal Impact of LN in Early Layers

Building on the observation that LN has distinctive impacts on learning & memorization for Pre- and Post-LN models, respectively, we now precisely investigate the impact of the early, middle, and later LN layers on Pre- and Post-LN models. Fig. 3 depicts that early LNs are more significant than middle/later LNs in driving learning and memorization in both Pre- and Post-LN models.

In the Pre-LN model (DeiT, Fig. 3a), the removal of early LNs parameters significantly disrupts the learning process, highlighting their importance in learning for Pre-LN models. In the Post-LN model



(a) Impact of early, middle, later LNs on learning (test) accuracy, memorization, recovery and random predicitons scores for Pre-LN models (DeiT, UTK-Face)



(b) Impact of early, middle, later LNs on learning (test) accuracy, memorization, recovery and random predicitons scores for Post-LN models (DeBERTa, Emotions)

Figure 3: **Pivotal impact of early LNs for learning and memorization across Pre- and Post-LN models.** (a) clearly shows impact of early LNs removal on destabilizing learning in Pre-LN models, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than later layers, whereas (b) shows early LNs removal help in suppressing memorization and improving recovery in Post-LN models, alongwith lower train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Post, early}}$, than later layers.

(DeBERTa, Fig. 3b), the removal of early LNs parameters mitigates memorization and enhances true label recovery most significantly compared to the cases of middle or later layers. This contrast highlights the pivotal impact of early LNs in shaping learning and memorization dynamics, positively in Post-LN and negatively in Pre-LN models. Aligned trends are observed for other multiple Pre- and Post-LN models, as presented in Appendix G.2. Prior studies [Gromov et al., 2024, Li et al., 2024, Lad et al., 2024, Men et al., 2024] highlighted the limited effectiveness of deeper layers for learning in Pre-LN models. Our observations take this a step further by precisely identifying that **LN in the early layers** is a critical factor in **memorization in Post-LN models**, presenting a *novel* and *distinctive* observation.

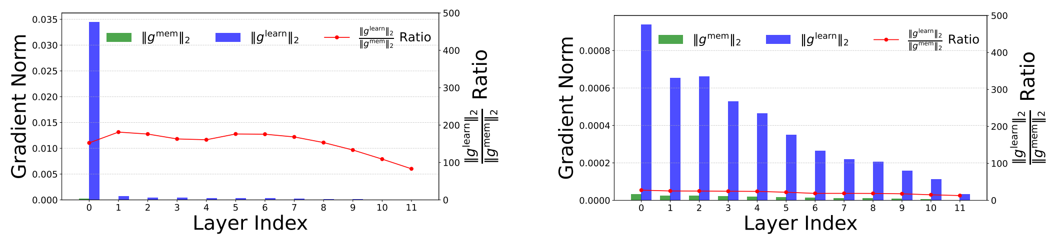
The distinctive effect of early LNs parameters removal—disrupting learning in Pre-LN models while mitigating memorization in Post-LN models—is further supported by the train-test accuracy gap (Δ_{overfit}). Specifically, in Pre-LN models, removing them leads to a more pronounced increase in $\Delta_{\text{overfit}}^{\text{Pre, early}}$ compared to middle or later LNs, whereas in Post-LN models, removing early LNs parameters results in a sharper decrease in $\Delta_{\text{overfit}}^{\text{Post, early}}$. This trend is shown in Fig. 3 (bar plots) and formalized as follows:

$$\Delta_{\text{overfit}}^{\text{Pre, early}} > \Delta_{\text{overfit}}^{\text{Pre, middle}} > \Delta_{\text{overfit}}^{\text{Pre, later}}, \quad \text{and} \quad \Delta_{\text{overfit}}^{\text{Post, early}} < \Delta_{\text{overfit}}^{\text{Post, middle}} < \Delta_{\text{overfit}}^{\text{Post, later}} \tag{3}$$

In summary, we observe that early layers LN are more significant than later layers LN, where their removal disrupts learning, explained by high $\Delta_{\text{overfit}}^{\text{Pre, early}}$ in Pre-LN models. On the other hand, their removal suppresses memorization while recovering true labels in Post-LN models, illustrated by low $\Delta_{\text{overfit}}^{\text{Post, early}}$. Similar trends are observed in other Pre-LN (GPTNeo, Qwen2, GPT2, ViT-B, ViT-S) and Post-LN (BERT, RoBERTa, ELECTRA, Longformer) models as illustrated in Appendix G.2.

6 Gradients Explain LN's Impact

To better understand the role of layer normalization (LN) in learning and memorization, we compute the gradient norms associated with both processes across different layers (g_x). Specifically, we measure the norms for learning (g_x^{learn}) and memorization (g_x^{mem}) gradients separately, allowing us to quantify their relative contributions throughout the network.



(a) Pre-LN (GPTNeo) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

(b) Post-LN (DeBERTa) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

Figure 4: **Learning vs. Memorization Gradients in Pre- and Post-LN Models:** (in Emotions Dataset) Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (GPTNeo) and Post-LN (DeBERTa) models. Importantly, the learning gradient norm ($\left\|g_x^{\text{learn}}\right\|_2$) is consistently stronger than the memorization gradient norm ($\|g_x^{\text{mem}}\|_2$) across all layers. Furthermore, the ratio $\left\|g_x^{\text{learn}}\right\|_2 / \|g_x^{\text{mem}}\|_2$ is significantly higher in Pre-LN models compared to Post-LN models.

Theorem 1 (Learning Gradient Norm, $\|\boldsymbol{g}_{\boldsymbol{x}}^{\textbf{learn}}\|_2$ is greater than or equal to Memorization Gradient Norm, $\|\boldsymbol{g}_{\boldsymbol{x}}^{\textbf{mem}}\|_2$ across all layers). *It is formally represented as follows:*

$$\left\|g_x^{\text{learn}}\right\|_2 \geq \|g_x^{\text{mem}}\|_2\,, \qquad \text{across all layers} \tag{4}$$

A proof of Theorem 1 is provided in Appendix B.

From Theorem 1, we observe that the learning gradient norms are generally greater than the memorizing gradient norms across all layers for both Pre- and Post-LN models. This observation is further validated empirically from the trend as seen in Fig. 4.

6.1 Understanding the Distinctive Impact of LN in Pre and Post-LN Architectures

Having identified the importance of early layers LNs (Sec. 5), we now focus on explaining why the removal of Layer Normalization (LN) in Pre-LN models hinders learning, while in Post-LN models, it mitigates memorization without disrupting learning. To do so, we focus on the ratio of learning-to-memorization gradients norms ($\frac{\|g_x^{\text{learn}}\|_2}{\|g_x^{\text{mem}}\|_2}$, red-color line plots in Fig. 4a & 4b) across layers. Based on the results, we uncover the following phenomenon:

$$\frac{\|g_x^{\text{learn}}\|_2}{\|g_x^{\text{mem}}\|_2}\Big|_{\text{Pre-LN}} \gg \frac{\|g_x^{\text{learn}}\|_2}{\|g_x^{\text{mem}}\|_2}\Big|_{\text{Post-LN}}, \qquad \text{across all layers} \tag{5}$$

This indicates that in Pre-LN models, LayerNorm primarily facilitates learning, as evidenced by the dominance of $\|g_x^{\text{learn}}\|_2$ over $\|g_x^{\text{mem}}\|_2$. Consequently, the removal of its parameters disrupts learning and exacerbates overfitting. In contrast, in Post-LN models, $\|g_x^{\text{learn}}\|_2$ and $\|g_x^{\text{mem}}\|_2$ are of comparable magnitudes. As a result, removing LN parameters effectively mitigates memorization by restoring genuine labels without disturbing learning. Consistent trends are observed for other Pre-LN (GPT2, Qwen2, ViT-B, DeiT, ViT-S) and Post-LN (RoBERTa, BERT, Longformer, ELECTRA) models, illustrated in Appendix G.3.

6.2 Why are LNs in Early Layers Important for Memorization and Learning?

In this section, we explain why the early layers LN are pivotal in governing memorization and learning across Post and Pre-LN models, through the lens of gradient analysis.

Theorem 2 (**Gradient norm of loss $\mathcal{L}$ w.r.t input of LN is upper bounded**).
Post-LN: Let z_i denote the input to LN$_1$ of the i^{th} Post-LN model layer. Then,

$$\|g_{z_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \leq s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N} \left|1 - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right| \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|} \right) \cdot \\ \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{x'_j})\right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right) \tag{6}$$

Pre-LN: Let x_i denote the input to LN$_1$ of the i^{th} Pre-LN model layer. Then,

$$\|g_{x_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial x_i}\right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} J_{\text{LN}_2}^{x'_j})\right) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{7}$$

A proof of Theorem 2, along with the expressions for LN$_2$ in both Pre- and Post-LN setup, is provided in Appendix C.

Theorem 3 (**Upper bound of the gradient norm of Early Layers LN are higher than those of Later layers LN**). *It is formally represented as follows:*

$$\text{UB}(\|g_{x_1}\|_2) \geq \text{UB}(\|g_{x_2}\|_2) \geq \cdots \geq \text{UB}(\|g_{x_N}\|_2) \text{ ; for both Pre- and Post-LN models} \tag{8}$$

where UB($\|g_{x_i}\|_2$) denotes the upper bound of $\|g_{x_i}\|_2$, and x_i is the input to the i^{th} layer's LN.

A proof of Theorem 3 is provided in Appendix D.

The results depicted in Fig.4 empirically confirm the trend established in Theorem 3. Specifically, we observe that both $\|g_x^{\text{learn}}\|_2$ and $\|g_x^{\text{mem}}\|_2$ are significantly higher in the earlier layers compared to the later ones. This gradient decay trend is consistent across both Pre-LN (GPTNeo, Fig.4a) and Post-LN (DeBERTa, Fig.4b) architectures. Similar trends are observed for other Pre-LN (GPT2, Qwen2, ViT-B, DeiT, ViT-S) and Post-LN (RoBERTa, BERT, Longformer, ELECTRA) models, as shown in Appendix G.3.

Thus, the theoretical upper bounds not only provide an analytical explanation for the gradient magnitude behavior but also align closely with the empirical patterns observed across a wide range of transformer variants. This alignment helps explain why the removal of early layers LN parameters leads to disruption of learning in Pre-LN models and mitigation of memorization in Post-LN models, highlighting their predominant role in the entire network because of their higher gradient norm.

In addition to the isolation of learning and memorization in early layers, we observe another interesting pattern. For Post-LN models (Fig. 4b), both $\|g_x^{\text{learn}}\|_2$ and $\|g_x^{\text{mem}}\|_2$ decrease gradually over layers LN. However, for Pre-LN models (Fig. 4a), the gradient norms are predominantly high in the first layer, with the following layers having almost negligible norms. This observation explains why the removal of early layers LN parameters did not significantly affect learning for Post-LN models. That is because, in Post-LN models, later LNs can compensate for the absence of the early ones, recovering learning, while mitigating memorization, due to their comparable gradient norms. However, this does not hold for Pre-LN models, where the high gradient norms in the early layers LN are critical, and their absence severely disrupts learning.

Similar observations are observed in other Pre-LN (GPT2, Qwen2, ViT-B, DeiT, ViT-S) and Post-LN (RoBERTa, BERT, Longformer, ELECTRA) models, as shown in Appendix G.3.

In summary, gradient analysis highlights why removal of LN parameters significantly affects both learning and memorization: (1) **disrupts learning in Pre-LN models and mitigates memorization in Post-LN models** due to the distinct behavior of their gradient norms ratio; and (2) **it reveals the particular significance of early layer LNs**, which exhibit stronger gradient norms and thus play a more influential role in both learning and memorization processes.

7 Conclusion

In conclusion, our study highlights the pivotal role of Layer Normalization (LN) in governing both memorization and learning across two different Transformer configurations: Pre-LN and Post-LN. We identified that the removal of LN parameters in Pre-LN models significantly destabilizes the learning process, leading to persistent overfitting. In contrast, removing LN parameters from Post-LN architectures effectively mitigates memorization and enables the recovery of genuine labels. More precisely, we find that LNs in the early layers are especially critical—removing them has the strongest impact in disrupting learning in Pre-LN models and mitigating memorization in Post-LN models. By analyzing the learning and memorization gradient norms, we further reveal how LN distinctively influences these two mechanisms across Pre- and Post-LN models. We show that this distinctive behavior across a wide range of model architectures, spanning several vision and language datasets. Overall, our findings uncover a crucial connection on how layer normalization impacts learning and memorization in transformer models, with its broader impacts discussed in Appendix K.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *ArXiv e-prints*, pages arXiv–1607, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *International Conference on Machine Learning*, 2023.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 248–264, 2023.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

Qinan Yu, Jack Merullo, and Ellie Pavlick. Characterizing mechanisms for factual recall in language models. *arXiv preprint arXiv:2310.15910*, 2023.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36:75067–75096, 2023.

Bobby He, Lorenzo Noci, Daniele Paliotta, Imanol Schlag, and Thomas Hofmann. Understanding and minimising outlier features in transformer training. *Advances in Neural Information Processing Systems*, 37:83786–83846, 2024.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

Xiaoling Zhou and Ou Wu. Which samples should be learned first: Easy or hard? *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. *arXiv preprint arXiv:2105.14602*, 2021.

Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2022.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. *arXiv preprint arXiv:2002.03206*, 2020.

Deepak Ravikumar, Efstathia Soufleri, Abolfazl Hashemi, and Kaushik Roy. Unveiling privacy, memorization, and input curvature links. *arXiv preprint arXiv:2402.18726*, 2024.

Isha Garg, Deepak Ravikumar, and Kaushik Roy. Memorization through the lens of curvature of loss function around samples. *arXiv preprint arXiv:2307.05831*, 2023.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*, 2024.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

Shaked Brody, Uri Alon, and Eran Yahav. On the expressivity role of layernorm in transformers' attention. *arXiv preprint arXiv:2305.02582*, 2023.

Xinyi Wu, Amir Ajorlou, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. On the role of attention masks and layernorm in transformers. *arXiv preprint arXiv:2405.18781*, 2024.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. B2t connection: Serving stability and performance in deep transformers. *arXiv preprint arXiv:2206.00330*, 2022.

Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. Residual: Transformer with dual residual connections. *arXiv preprint arXiv:2304.14802*, 2023.

Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting layer normalization in the transformer architecture. *arXiv preprint arXiv:2502.02732*, 2025.

Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. *arXiv preprint arXiv:2110.09456*, 2021.

Hongyu Wang, Shuming Ma, Shaohan Huang, Li Dong, Wenhui Wang, Zhiliang Peng, Yu Wu, Payal Bajaj, Saksham Singhal, Alon Benhaim, et al. Foundation transformers. *arXiv preprint arXiv:2210.06423*, 2022.

Manoj Kumar, Mostafa Dehghani, and Neil Houlsby. Dual patchnorm. *arXiv preprint arXiv:2302.01327*, 2023.

Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. Lipsformer: Introducing lipschitz continuity to vision transformers. *arXiv preprint arXiv:2304.09856*, 2023.

Zixuan Jiang, Jiaqi Gu, Hanqing Zhu, and David Pan. Pre-rmsnorm and pre-crmsnorm transformers: equivalent and efficient pre-ln transformers. *Advances in Neural Information Processing Systems*, 36:45777–45793, 2023.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Xingxuan Zhang, Yue He, Renzhe Xu, Han Yu, Zheyan Shen, and Peng Cui. Nico++: Towards better benchmarking for domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16036–16047, 2023.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL `https://aclanthology.org/D18-1404/`.

Okite97. News data dataset. `https://huggingface.co/datasets/okite97/news-data`, 2024. Accessed: 2024-03-03.

Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Leonardo Neves, Vitor Silva, and Francesco Barbieri. Twitter Topic Classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Liu Yinhan, Ott Myle, Goyal Naman, Du Jingfei, Joshi Mandar, Chen Danqi, Levy Omer, and Lewis Mike. Roberta: A robustly optimized bert pretraining approach (2019). *arXiv preprint arXiv:1907.11692*, pages 1–13, 2019.

Victor Sanh, L Debut, J Chaumond, and T Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arxiv 2019. *arXiv preprint arXiv:1910.01108*, 2019.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

K Clark. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Dosovitskiy Alexey. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020.

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European conference on computer vision*, pages 456–473. Springer, 2022.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow, 2021. *URL: https://doi. org/10.5281/zenodo*, 5297715, 2022.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL `https://aclanthology.org/N19-4009/`.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

Ola Hössjer and Arvid Sjölander. Sharp lower and upper bounds for the covariance of bounded random variables. *Statistics & Probability Letters*, 182:109323, 2022.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Appendix

Table 2: Comparison of Post-Layer Normalization (Post-LN) and Pre-Layer Normalization (Pre-LN) Transformer Setup. MHSA = multi-head self attention, FFN = feed-forward network, LN: LayerNorm, N = number of transformer layers, x_i = input to i^{th} layer, y_i = output of i^{th} layer, z_i & $z_{i'}$ = intermediate vectors in Post-LN model, which are inputs to LN_1 & LN_2 respectively.

Post-LN Transformer Setup	**Pre-LN Transformer Setup**
$z_i = \text{MHSA}(x_i) + x_i$ $x'_i = \text{LN}_1(z_i)$ $z'_i = \text{FFN}(x'_i) + x'_i$ $y_i = \text{LN}_2(z'_i)$	$x'_i = \text{MHSA}(\text{LN}_1(x_i)) + x_i$ $y_i = \text{FFN}(\text{LN}_2(x'_i)) + x'_i$
$y_{\text{out}} = \text{classification-head}(y_N)$ $\mathcal{L} = \text{CrossEntropyloss}(y_{\text{out}}, y_{\text{true}})$	

A Problem Setup for Gradients Analysis:

Consider a training dataset D_{train} consisting of C classes. We make an assumption that all samples in class c well represent class c. Then, we introduce a single noisy label by selecting a sample (x_1, y_c) from class c, where $c \in C$, and modify its label to a different (incorrect) label, y_{NL}, where $\text{NL}(\in C) \neq c$. The noisy sample is now represented as (x_1, y_{NL}). At the same time, we do not modify any other training samples from class c to ensure that the model has access to correctly labeled examples for effective learning as well. Next, we train a transformer model on this modified dataset until it reaches 100% training accuracy. At this stage, the model has fully memorized the noisy-labeled sample (x_1, y_{NL}) while also learning class c features from correctly labeled training samples. Now to measure the notion of memorization and learning, we compute ***memorization gradient norm*** ($\|g_x^{\text{mem}}\|_2$) and ***learning gradient norm*** ($\|g_x^{\text{learn}}\|_2$) as discussed in Sec.3.2, and compare them.

B Theorem 1: Learning Gradient Norm, $\|\boldsymbol{g}_{\boldsymbol{x}}^{\text{learn}}\|_2$ is greater than or equal to Memorization Gradient Norm, $\|\boldsymbol{g}_{\boldsymbol{x}}^{\text{mem}}\|_2$ across all layers.

It is formally represented as follows:

$$\left\|g_x^{\text{learn}}\right\|_2 \geq \|g_x^{\text{mem}}\|_2\,, \qquad \text{across all layers} \tag{9}$$

Proof:

Based on the Problem Setup as discussed in Sec. A, we prove that $\left\|g_x^{\text{learn}}\right\|_2 \geq \|g_x^{\text{mem}}\|_2$ across all layers, for both Pre- and Post-LN models as follows:

B.1 *For Post-LN model:*

Firstly, we elucidate the architecture of Post-LN transformer in Tab. 2. Based on the Post-LN architecture, during backpropagation, we compute derivatives of loss w.r.t input of LN for every i^{th}-layer. Since there are two LNs in every layer, we compute and compare the learning and memorization gradients corresponding to the inputs of both LNs, i.e., $g_{z_{i'}}$ and g_{z_i} (refer to Tab. 2).

B.1.1 Backpropagating gradient analysis for LN_2 $(g_{z'_i})$:

The backpropagating gradient for $g_{z_{i'}}$ can be expressed as follows:

$$g_{z'_i} = \frac{\partial \mathcal{L}}{\partial z'_i} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z'_j} \cdot \frac{\partial z'_j}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial z'_i} \tag{10}$$

where $x_{i+1} = y_i$ because i^{th} layer's output y_i is the input of $(i+1)^{\text{th}}$ layer, x_{i+1}. To measure memorization and learning, we compute these gradients for both (x_1, y_{NL}) and (x_2, y_c), denoted as

$g_{z_{i'}}^{\text{mem}}$ and $g_{z_{i'}}^{\text{learn}}$, respectively. In both gradients, $y_{\text{out}}, y_j, z'_j, x'_j, z_j, x_j$ are only dependent on the input samples x_1 and x_2, respectively, and not on their labels, where both x_1 and x_2 genuinely represent class c. Therefore, $\frac{\partial y_{\text{out}}}{\partial y_N}, \frac{\partial y_j}{\partial z'_j}, \frac{\partial z'_j}{\partial x'_j}, \frac{\partial x'_j}{\partial z_j}, \frac{\partial z_j}{\partial x_j}, \frac{\partial y_i}{\partial z_i}$ will not be significantly different for both g_x^{mem} and g_x^{learn}, thus we regard the term, $\frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z'_j} \cdot \frac{\partial z'_j}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial z'_i}$, as A_1 for both cases (as it does not vary across inputs). The only difference between the two gradients is due to $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}}$, because the loss $\mathcal{L}$ is dependent on the label of both samples, i.e., y_c and y_{NL}, as follows:

$$\mathcal{L} = -\sum_{k_i=1}^{C} y_{k_i} \log(\hat{y}_{k_i}) \tag{11}$$

where $y_{k_i} = 1$ if k_i is the ground truth class, otherwise 0, and $\hat{y}_{k_i}$ is the predicted softmax probability of class k_i. As a result, $g_{z_{i'}}^{\text{mem}}$ and $g_{z_{i'}}^{\text{learn}}$ can be respectively represented as follows:

$$g_{z_{i'}}^{\text{mem}} = \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \cdot A_1 \quad \& \quad g_{z_{i'}}^{\text{learn}} = \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \cdot A_1 \tag{12}$$

Comparing learning and memorization gradients Now the problem boils down to comparing the L2-norms of $\frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}}$ and $\frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}}$. To do so, we need to first define $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}}$ We know that $\mathcal{L}$ is the CrossEntropyLoss between the predicted softmax probability vector $\hat{y} = \text{Softmax}(y_{out})$ and the ground truth y as defined in Eq. (11). Hence, $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}}$ can be written as follows:

$$\frac{\partial \mathcal{L}}{\partial y_{\text{out}}} = \hat{y} - y \tag{13}$$

Now we can understand what it really means for memorization and learning. During inference of (x_1, y_{NL}), $\hat{y_{\text{mem}}}$ will be a vector which consists of very high probability (≈ 1) for class y_{NL} while assigning extremely low probabilities (≈ 0) to remaining classes due to overfitting. Therefore, $\hat{y_{\text{mem}}} - y_{\text{NL}}$ will almost be a 0-vector, i.e., all the elements in the vector would be almost 0. This phenomenon is formally represented as follows:

$$\begin{aligned} \hat{y}_{\text{mem}} &\approx [0, \ldots, 1, \ldots, 0], \\ y_{\text{NL}} &= [0, \ldots, 1, \ldots, 0], \\ \hat{y}_{\text{mem}} - y_{\text{NL}} &\approx [0, \ldots, 0, \ldots, 0]. \end{aligned} \tag{14}$$

where the index corresponding to class y_{NL} is 1, and all other elements are (close to) 0.

Now, we can take the L2-norm of both sides in Eq. (13) for the memorizing sample. Since $\hat{y_{\text{mem}}} - y_{\text{NL}}$ is close to a 0-vector, its L2-norm ≈ 0. Therefore,

$$\left\| \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \right\|_2 \approx 0 \tag{15}$$

On the other hand, for (x_1, y_{c}), even though the model has learned generalizable features for class c, it has also been overfitted on noisy label samples. Hence, $\hat{y_{\text{learn}}}$ will not assign a very high probability to y_c but instead will distribute some probability mass across multiple classes. Therefore, $\hat{y_{\text{learn}}} - y_{\text{c}}$ contains non-trivial, not near-zero values. This behavior is formally presented as follows:

$$\begin{aligned} \hat{y}_{\text{learn}} &= [p_1, p_2, \ldots, p_{y_c}, \ldots, p_C], \\ y_c &= [0, \ldots, 1, \ldots, 0], \\ \hat{y}_{\text{learn}} - y_c &= [p_1, p_2, \ldots, p_{y_c} - 1, \ldots, p_C]. \end{aligned} \tag{16}$$

where p_{y_c} is not close to 1, and other probabilities p_i are non-negligible.

Now, by taking the L2-norm on both sides of Eq. (13) for the learning sample. Since $\hat{y_{\text{learn}}} - y_{\text{c}}$ contains non-trivial, non zero values, its L2-norm $\gg 0$. Therefore,

$$\left\| \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \right\|_2 \gg 0 \tag{17}$$

Therefore, by comparing Eq. (15) **&** (17)**, we can establish the following relationship:**

$$\left\| \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \right\|_2 \geq \left\| \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \right\|_2, \tag{18}$$

because overfitting on noisy samples, causes the memorizing gradients to be smaller than learning gradients. However, note that an ideal case of perfect learning, where 100% memorization and 100% learning co-exist, is not achievable in practice as memorization inherently hinders generalization. Hence, the equality from the inequality can be disregarded in almost all cases.

Now, substituting the relation found in Eq. (18) to the L2-norms of $g_{z_i'}^{\text{learn}}$ and $g_{z_i'}^{\text{mem}}$ in Eq. (12), we can formally conclude that:

$$\left\| g_{z_i'}^{\text{learn}} \right\|_2 \geq \left\| g_{z_i'}^{\text{mem}} \right\|_2 \tag{19}$$

This proof explains why $\left\| g_{z_i'}^{\text{mem}} \right\|_2$ is lower than $\left\| g_{z_i'}^{\text{learn}} \right\|_2$ across all layers, as also empirically observed in Fig. 4.

B.1.2 Backpropagating gradient analysis for LN$_1$ (g_{z_i}):

Similar to $g_{z_i'}$, we can express g_{z_i} as follows:

$$g_{z_i} = \frac{\partial \mathcal{L}}{\partial z_i} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z_j'} \cdot \frac{\partial z_j'}{\partial x_j'} \cdot \frac{\partial x_j'}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial z_i'} \cdot \frac{\partial z_i'}{\partial x_i'} \cdot \frac{\partial x_i'}{\partial z_i} \tag{20}$$

Here, $x_{i+1} = y_i$ because i^{th} layer's output y_i is the input for $(i+1)^{\text{th}}$ layer, x_{i+1}.

We compute $g_{z_i}^{\text{mem}}$ and $g_{z_i}^{\text{learn}}$ for both memorization (x_1, y_{NL}) and learning (x_2, y_{c}) samples, respectively. Based on the discussion in Sec. B.1.1 on the similarity of x_1 and x_2, since they both originally belong to the same class c, we can write $g_{z_{i'}}^{\text{mem}}$ and $g_{z_{i'}}^{\text{learn}}$ as follows:

$$g_{z_i}^{\text{mem}} = \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \cdot A_2 \quad \& \quad g_{z_i}^{\text{learn}} = \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \cdot A_2 \tag{21}$$

where, $A_2 = \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z_j'} \cdot \frac{\partial z_j'}{\partial x_j'} \cdot \frac{\partial x_j'}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial z_i'} \cdot \frac{\partial z_i'}{\partial x_i'} \cdot \frac{\partial x_i'}{\partial z_i}$, which does not vary across inputs.

To compare $g_{z_i}^{\text{mem}}$ and $g_{z_i}^{\text{learn}}$, we use the argument made in Eq. (14) & (16), which explains why $\left\| \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \right\|_2 \geq \left\| \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \right\|_2$.

Using the above results and subsituting the relation in the L2-norms of $g_{z_i}^{\text{learn}}$ and $g_{z_i}^{\text{mem}}$ in Eq. (21), we can conclude that:

$$\left\| g_{z_i}^{\text{learn}} \right\|_2 \geq \left\| g_{z_i}^{\text{mem}} \right\|_2 \tag{22}$$

In conclusion, both Eq. (19) & (22) formally demonstrate that the L2-norm of ***learning gradient***, g_x^{learn} is greater than or equal to ***memorization gradient***, g_x^{mem} across all layers of a Post-LN model.

B.2 *For Pre-LN model:*

Firstly, we describe the architecture of the Pre-LN transformer in Tab. 2. Based on the Pre-LN architecture, during backpropagation, we compute derivatives of loss wrt input of LN for every i^{th} layer. Since there are two LNs in every layer, we compute and compare the learning and memorization gradients corresponding to the inputs of both LNs, i.e., $g_{x_{i'}}$ and g_{x_i}.

B.2.1 Backpropagating gradient analysis for LN$_2$ $(g_{x_i'})$:

The backpropagating gradient for $g_{x_{i'}}$ can be expressed as follows:

$$g_{x_i'} = \frac{\partial \mathcal{L}}{\partial x_i'} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial x_j'} \cdot \frac{\partial x_{j'}}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial x_i'} \tag{23}$$

where $x_{i+1} = y_i$ because i^{th} layer's output y_i is the input of $(i+1)^{\text{th}}$ layer, x_{i+1}. To measure memorization and learning, we then compute these gradients for both (x_1, y_{NL}) and (x_2, y_c). In both gradients, $y_{\text{out}}, y_j, x_j', x_j$ are only dependent on the input samples x_1 and x_2, respectively, and not on their labels, where both x_1 and x_2 genuinely represent class c. Therefore, $\frac{\partial y_{\text{out}}}{\partial y_N}, \frac{\partial y_j}{\partial x_j'}, \frac{\partial x_j'}{\partial x_j}, \frac{\partial y_i}{\partial x_i'}$ will not be significantly different for both g_x^{learn} and g_x^{mem}, thus we regard the term, $\frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial x_j'} \cdot \frac{\partial x_j'}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial x_i'}$, as B_1 for both cases (as it does not vary across inputs.) The only difference between the two gradients is due to $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}}$, because the loss $\mathcal{L}$ is dependent on the label of both samples, i.e., y_c and y_{NL} as shown in Eq. (11). As a result, $g_{x_{i'}}^{mem}$ and $g_{x_{i'}}^{learn}$ can be respectively represented as follows:

$$g_{x_{i'}}^{\text{mem}} = \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \cdot B_1 \quad \& \quad g_{x_{i'}}^{\text{learn}} = \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \cdot B_1 \tag{24}$$

Comparing learning and memorization gradient norms Now the problem boils down to comparing the norms of $\frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}}$ and $\frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}}$.

From Eq. (18)**, we know the following relation:**

$$\left\| \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \right\|_2 \geq \left\| \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \right\|_2, \tag{25}$$

Now, substituting the relation found in Eq. (18) to the l2-norms of $g_{x_i'}^{\text{learn}}$ and $g_{x_i'}^{\text{mem}}$ in Eq. (24), we can formally conclude that:

$$\left\| g_{x_i'}^{\text{learn}} \right\|_2 \geq \left\| g_{x_i'}^{\text{mem}} \right\|_2 \tag{26}$$

This proof explains why $\left\| g_{x_i'}^{\text{mem}} \right\|_2$ is lower than $\left\| g_{x_i'}^{\text{learn}} \right\|_2$ across all layers, as also empirically consistently observed in Fig. 4.

B.2.2 Backpropagating gradient analysis for LN$_1$ (g_{x_i}):

Similar to $g_{x_{i'}}$, we can express g_{x_i} as follows:

$$g_{x_i} = \frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial x_j'} \cdot \frac{\partial x_{j'}}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial x_i'} \cdot \frac{\partial x_i'}{\partial x_i} \tag{27}$$

where $x_{i+1} = y_i$ because i^{th} layer's output y_i is the input for $(i+1)^{\text{th}}$ layer, x_{i+1}, and compute $g_{z_i}^{\text{mem}}$ and $g_{z_i}^{\text{learn}}$ to measure memorization and learning respectively. Based on the discussion in Sec. B.1.1 on the similarity of x_1 and x_2 since they both genuinely belong to the same class c, we can write $g_{x_i}^{mem}$ and $g_{x_i}^{learn}$ as follows:

$$g_{x_i}^{\text{mem}} = \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \cdot B_2 \quad \& \quad g_{x_i}^{\text{learn}} = \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \cdot B_2 \tag{28}$$

where, $B_2 = \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial x_j'} \cdot \frac{\partial x_{j'}}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial x_i'} \cdot \frac{\partial x_i'}{\partial x_i}$, which does not vary across inputs. To compare $g_{x_i}^{\text{mem}}$ and $g_{x_i}^{\text{learn}}$, we use Eq. (18), which states that $\left\| \frac{\partial \mathcal{L}^{\text{learn}}}{\partial y_{\text{out}}^{\text{learn}}} \right\|_2 \geq \left\| \frac{\partial \mathcal{L}^{\text{mem}}}{\partial y_{\text{out}}^{\text{mem}}} \right\|_2$.

Using the above results and substituting the relation in the L2-norms of $g_{x_i}^{\text{learn}}$ and $g_{x_i}^{\text{mem}}$ in Eq. (28), we can conclude that:

$$\left\|g_{x_i}^{\text{learn}}\right\|_2 \geq \left\|g_{x_i}^{\text{mem}}\right\|_2 \tag{29}$$

In conclusion, both Eq. (26) & (29) formally demonstrate that the L2-norm of ***learning gradient***, g_x^{learn} is greater than or equal to ***memorization gradient***, g_x^{mem} across all layers of a Pre-LN model.

□

C Theorem 2: Gradient norm of loss $\mathcal{L}$ w.r.t input of LN is upper bounded.

Post-LN: Let z_i denote the input to LN$_1$ of the i^{th} Post-LN model layer. Then,

$$\|g_{z_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \leq s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N}\left|1-\sqrt{\text{Var}(\text{FFN}(x_j'))}\right|\left|1-\sqrt{\text{Var}(\text{MHSA}(x_j))}\right|}\right) \cdot \prod_{j=i}^{N}\left(1+s_{\max}(J_{\text{FFN}}^{x_j'})\right) \cdot \prod_{j=i+1}^{N}\left(1+s_{\max}(J_{\text{MHSA}}^{x_j})\right) \tag{30}$$

Pre-LN: Let x_i denote the input to LN$_1$ of the i^{th} Pre-LN model layer. Then,

$$\|g_{x_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial x_i}\right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N}\left((1+s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x_j')} J_{\text{LN}_2}^{x_j'})\right) \cdot \prod_{j=i}^{N}\left((1+s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{31}$$

Proof:

C.1 For Post-LN model:

The Post-LN model setup for i^{th} layer can be represented as follows:

$$\begin{aligned} x_i' &= \text{LN}_1(x_i + \text{MHSA}(x_i)) \\ y_i &= \text{LN}_2(x_i' + \text{FFN}(x_i')) \end{aligned} \tag{32}$$

where $x_i + \text{MHSA}(x_i)$ and $x_i' + \text{FFN}(x_i')$ are the inputs to LN$_1$ and LN$_2$, respectively. Later, we substitute and use them as

$$\begin{aligned} z_i &= x_i + \text{MHSA}(x_i) \\ z_{i'} &= x_i' + \text{FFN}(x_i') \end{aligned} \tag{33}$$

Since there are two LayerNorm (LN) operations in every layer, we prove it seperately for both of them.

C.1.1 Backpropagation analysis for LN$_2$ ($g_{z_i'}$):

By applying Eq. (33), we obtain $z_j' = x_j' + \text{FFN}(x_j')$, $z_j = x_j + \text{MHSA}(x_j)$. Hence, we can write $g_{z_i'}$ (from Eq. (10)) for the i^{th} layer as follows:

$$g_{z_i'} = \frac{\partial \mathcal{L}}{\partial z_i'} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N}\left(\frac{\partial y_j}{\partial z_j'} \cdot \frac{\partial z_j'}{\partial x_j'} \cdot \frac{\partial x_j'}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j}\right) \cdot \frac{\partial y_i}{\partial z_i'} \tag{34}$$

Here, $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N}$, is independent of the transformer's layers as they are computed using the classification head's output. Therefore, we can treat them as P_1 (which does not vary across layers). We also compute the corresponding derivatives of $z_{j'}$ and z_j. Lastly, x_{i+1} is same as y_i, because y_i is the output of the i^{th} layer which becomes input x_{i+1} of the $(i+1)^{\text{th}}$ layer. By applying all of these, we obtain the following equation:

$$g_{z'_i} = \frac{\partial \mathcal{L}}{\partial z'_i} = P_1 \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z'_j} \cdot \frac{\partial (x'_j + \text{FFN}(x_{j'}))}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial z_j} \cdot \frac{\partial (x_j + \text{MHSA}(x_j))}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial z'_i} \tag{35}$$

$$= P_1 \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial z'_j} \cdot (\mathrm{I} + \frac{\partial \text{FFN}(x'_j)}{\partial x'_j}) \cdot \frac{\partial x'_j}{\partial z_j} \cdot (\mathrm{I} + \frac{\partial \text{MHSA}(x_j)}{\partial x_j}) \right) \cdot \frac{\partial y_i}{\partial z'_i} \tag{36}$$

We also acknowledge that $\frac{\partial y_j}{\partial z_j}$, $\frac{\partial x'_j}{\partial z'_j}$, and $\frac{\partial y_i}{\partial z'_i}$, are derivatives of output of LN w.r.t their input, which can be simply represented as Jacobian matrices, $J^{z_j}_{\text{LN}_2}$, $J^{z'_j}_{\text{LN}_1}$, and $J^{z'_i}_{\text{LN}_2}$, respectively. Likewise, $\frac{\partial \text{FFN}(x'_j)}{\partial x'_j}$ and $\frac{\partial \text{MHSA}(x_j)}{\partial x_j}$ are also derivatives of the output of FFN and MHSA w.r.t their inputs, and can be represented as Jacobian matrices, $J^{x'_j}_{\text{FFN}}$ and $J^{x_j}_{\text{MHSA}}$, respectively. After substituting these terms, we obtain the following:

$$g_{z'_i} = \frac{\partial \mathcal{L}}{\partial z'_i} = P_1 \cdot \prod_{j=i+1}^{N} \left(J^{z'_j}_{\text{LN}_2} \cdot (\mathrm{I} + J^{x'_j}_{\text{FFN}}) \cdot J^{z_j}_{\text{LN}_1} \cdot (\mathrm{I} + J^{x_j}_{\text{MHSA}}) \right) \cdot J^{z'_i}_{\text{LN}_2} \tag{37}$$

Now, we take the L2-norm on both sides of Eq. (37) as follows:

$$\|g_{z'_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial z'_i} \right\|_2 = \left\| P_1 \cdot \prod_{j=i+1}^{N} \left(J^{z'_j}_{\text{LN}_2} \cdot (\mathrm{I} + J^{x'_j}_{\text{FFN}}) \cdot J^{z_j}_{\text{LN}_1} \cdot (\mathrm{I} + J^{x_j}_{\text{MHSA}}) \right) \cdot J^{z'_i}_{\text{LN}_2} \right\|_2 \tag{38}$$

We know that L2-norm of a matrix is equivalent to its largest singular value [Horn and Johnson, 1991]. Hence, we can further write Eq. (38) as follows:

$$\|g_{z'_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial z'_i} \right\|_2 = s_{\max}(P_1 \cdot \prod_{j=i+1}^{N} \left(J^{z'_j}_{\text{LN}_2} \cdot (\mathrm{I} + J^{x'_j}_{\text{FFN}}) \cdot J^{z_j}_{\text{LN}_1} \cdot (\mathrm{I} + J^{x_j}_{\text{MHSA}}) \right) \cdot J^{z'_i}_{\text{LN}_2}) \tag{39}$$

where $s_{\max}$ outputs the largest singular value of $(P_1 \cdot \prod_{j=i+1}^{N} \left(J^{z'_j}_{\text{LN}_2} \cdot (\mathrm{I} + J^{x'_j}_{\text{FFN}}) \cdot J^{z_j}_{\text{LN}_1} \cdot (\mathrm{I} + J^{x_j}_{\text{MHSA}}) \right) \cdot J^{z'_i}_{\text{LN}_2})$. From the properties of singular values [Horn and Johnson, 1991], we know that

$$\begin{aligned} s_{\max}(A_1 A_2 \dots A_n) &\leq s_{\max}(A_1) s_{\max}(A_2) \dots s_{\max}(A_n) \\ s_{\max}(A_1 + A_2) &\leq s_{\max}(A_1) + s_{\max}(A_2) \end{aligned} \tag{40}$$

where $s_{\max}(A_k)$ is the maximum singular value of matrix A_k. After applying these properties to Eq. (39), we get the following:

$$\begin{aligned} \|g_{z'_i}\|_2 &= \left\| \frac{\partial \mathcal{L}}{\partial z'_i} \right\|_2 \\ &\leq s_{\max}(P_1) \cdot \prod_{j=i+1}^{N} \Bigg(s_{\max}(J^{z'_j}_{\text{LN}_2}) \cdot \big(s_{\max}(\mathrm{I}) + s_{\max}(J^{x'_j}_{\text{FFN}}) \big) \cdot \\ &\qquad s_{\max}(J^{z_j}_{\text{LN}_1}) \cdot \big(s_{\max}(\mathrm{I}) + s_{\max}(J^{x_j}_{\text{MHSA}}) \big) \Bigg) \cdot s_{\max}(J^{z'_i}_{\text{LN}_2}) \end{aligned} \tag{41}$$

According to Xiong et al. [2020], we can rewrite the Jacobian of LNs as follows:

$$J^{z_j}_{\text{LN}_1} = \frac{\mathrm{I}}{\sigma_{z_j}}, \quad J^{z'_j}_{\text{LN}_2} = \frac{\mathrm{I}}{\sigma_{z'_j}}, \quad \textit{and} \quad J^{z'_i}_{\text{LN}_2} = \frac{\mathrm{I}}{\sigma_{z'_i}} \tag{42}$$

where σ_{z_j}, $\sigma_{z'_j}$, and $\sigma_{z'_i}$ are the standard-deviations of z_j, z'_j, and z'_i, respectively. Therefore, we obtain the following equation:

$$\begin{aligned}\|g_{z'_i}\|_2 &= \left\|\frac{\partial \mathcal{L}}{\partial z'_i}\right\|_2 \\ &\leq s_{\max}(P_1) \cdot \prod_{j=i+1}^{N} \left(s_{\max}\left(\frac{\mathrm{I}}{\sigma_{z'_j}}\right) \cdot \left(s_{\max}(\mathrm{I}) + s_{\max}(J_{\text{FFN}}^{x'_j})\right) \cdot \right. \\ &\qquad \left. s_{\max}\left(\frac{\mathrm{I}}{\sigma_{z_j}}\right) \cdot \left(s_{\max}(\mathrm{I}) + s_{\max}(J_{\text{MHSA}}^{x_j})\right)\right) \cdot s_{\max}\left(\frac{\mathrm{I}}{\sigma_{z'_i}}\right).\end{aligned} \tag{43}$$

Another property of singular values states that all singular values of identity matrix I are 1 [Horn and Johnson, 1991], i.e., $s_k(\mathrm{I}) = 1$. Therefore substituting with that in Eq. (43), we obtain the following:

$$\|g_{z'_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z'_i}\right\|_2 \leq s_{\max}(P_1) \cdot \prod_{j=i+1}^{N} \left(\frac{1}{\sigma_{z'_j}} \cdot (1 + s_{\max}(J_{\text{FFN}}^{x'_j})) \cdot \frac{1}{\sigma_{z_j}} \cdot (1 + s_{\max}(J_{\text{MHSA}}^{x_j}))\right) \cdot \frac{1}{\sigma_{z'_i}} \tag{44}$$

By re-arranging the terms, we finally obtain the following equation:

$$\begin{aligned}\|g_{z'_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z'_i}\right\|_2 &\leq s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N} \sigma_{z'_j}}\right) \cdot \left(\frac{1}{\prod_{j=i+1}^{N} \sigma_{z_j}}\right) \\ &\cdot \prod_{j=i+1}^{N} \left(\left(1 + s_{\max}(J_{\text{FFN}}^{x'_j})\right) \cdot \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right)\right)\end{aligned} \tag{45}$$

Now, we need to investigate how σ_{z_j} and $\sigma_{z_{j'}}$ behave. We know that $\sigma_{z_j} = \sqrt{\text{Var}(z_j)}$ and $\sigma_{z'_j} = \sqrt{\text{Var}(z'_j)}$, where $\text{Var}(\cdot)$ represents a variance. Therefore, we can instead focus on $\text{Var}(z_j)$ and $\text{Var}(z'_j)$. We know that $z_j = x_j + \text{MHSA}(x_j)$ and $z'_j = x'_j + \text{FFN}(x'_j)$. Therefore, their variances can be written as follows:

$$\begin{aligned}\text{Var}(z_j) &= \text{Var}(x_j + \text{MHSA}(x_j)) \\ \text{Var}(z'_j) &= \text{Var}(x'_j + \text{FFN}(x'_j))\end{aligned} \tag{46}$$

Now to compute the upper bound of Eq. (45), we need to substitute the lower bound of σ_{z_j} and $\sigma_{z'_j}$ as they are in the denominator. The lower bounds of σ_{z_j} and $\sigma_{z'_j}$ would basically be lower bounds of $\text{Var}(z_j)$ and $\text{Var}(z'_j)$, respectively.

From Hössjer and Sjölander [2022], we know that for any two matrices A and B,

$$\text{Var}(\mathrm{A} + \mathrm{B}) \geq \left(\sqrt{\text{Var}(\mathrm{A})} - \sqrt{\text{Var}(\mathrm{B})}\right)^2 \tag{47}$$

Therefore, the lower bounds of $\text{Var}(z_j)$ and $\text{Var}(z'_j)$ can be written as follows,

$$\begin{aligned}\text{Var}(z_j) &\geq \left(\sqrt{\text{Var}(x_j)} - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right)^2 \\ \text{Var}(z'_j) &\geq \left(\sqrt{\text{Var}(x'_j)} - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right)^2\end{aligned} \tag{48}$$

Since x_j and x'_j are the outputs of the two LN layers, their variance is 1. Therefore, we can rewrite Eq. (48) as follows:

$$\begin{aligned}\text{Var}(z_j) \geq \left(1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right)^2 \quad &\Rightarrow \quad \sigma_{z_j} \geq \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right| \\ \text{Var}(z'_j) \geq \left(1 - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right)^2 \quad &\Rightarrow \quad \sigma_{z'_j} \geq \left|1 - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right|\end{aligned} \tag{49}$$

Hence, we can re-write Eq. (45) as follows:

$$\|g_{z'_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z'_i}\right\|_2 \leq s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N}\left|1-\sqrt{\text{Var}(\text{FFN}(x'_j))}\right|}\right) \cdot \left(\frac{1}{\prod_{j=i+1}^{N}\left|1-\sqrt{\text{Var}(\text{MHSA}(x_j))}\right|}\right) \cdot \prod_{j=i+1}^{N}\left(\left(1+s_{\max}(J_{\text{FFN}}^{x'_j})\right)\cdot\left(1+s_{\max}(J_{\text{MHSA}}^{x_j})\right)\right) \tag{50}$$

C.1.2 Backpropagation analysis for LN$_1$ (g_{z_i}):

Similar to $g_{z'_i}$, we can express g_{z_i} as follows:

$$g_{z_i} = \frac{\partial \mathcal{L}}{\partial z_i} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=l+1}^{N}\left(\frac{\partial y_j}{\partial z'_j} \cdot \frac{\partial z'_j}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_j}\right) \cdot \frac{\partial y_i}{\partial z'_i} \cdot \frac{\partial z'_i}{\partial x'_i} \cdot \frac{\partial x'_i}{\partial z_i} \tag{51}$$

Here, $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N}$, is independent of the transformer's layers as they are computed using the classification head's output. Therefore, we can treat them as P_1 (which does not vary across layers). We also compute the corresponding derivatives of $z_{j'}$ and z_j. Lastly, x_{i+1} is same as y_i, because y_i is the output of the i^{th} layer which becomes input x_{i+1} of the $(i+1)^{\text{th}}$ layer. By applying all of these, we obtain the following equation:

$$g_{z_i} = \frac{\partial \mathcal{L}}{\partial z_i} = P_1 \cdot \prod_{j=i+1}^{N}\left(\frac{\partial y_j}{\partial z'_j} \cdot \frac{\partial(x'_j + \text{FFN}(x'_j))}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial z_j} \cdot \frac{\partial(x_j + \text{MHSA}(x_j))}{\partial x_j}\right) \cdot \frac{\partial y_i}{\partial z'_i} \cdot \frac{\partial(x'_i + \text{FFN}(x'_i))}{\partial x'_i} \cdot \frac{\partial x'_i}{\partial z_i} \tag{52}$$

$$= P_1 \cdot \prod_{j=i+1}^{N}\left(\frac{\partial y_j}{\partial z'_j} \cdot \left(\text{I} + \frac{\partial\,\text{FFN}(x'_j)}{\partial x'_j}\right) \cdot \frac{\partial x'_j}{\partial z_j} \cdot \left(\text{I} + \frac{\partial\,\text{MHSA}(x_j)}{\partial x_j}\right)\right) \cdot \frac{\partial y_i}{\partial z'_i} \cdot \left(\text{I} + \frac{\partial\,\text{FFN}(x'_i)}{\partial x'_i}\right) \cdot \frac{\partial x'_i}{\partial z_i} \tag{53}$$

Clearly, we can see that $\frac{\partial y_j}{\partial z_j}$, $\frac{\partial x'_j}{\partial z'_j}$, $\frac{\partial y_i}{\partial z'_i}$, and $\frac{\partial x'_i}{\partial z_i}$ are derivatives of output of LN w.r.t their input, which can be simply represented as Jacobian matrices, $J_{\text{LN}_2}^{z_j}$, $J_{\text{LN}_1}^{z'_j}$, $J_{\text{LN}_2}^{z'_i}$, and $J_{\text{LN}_1}^{z_i}$, respectively. Likewise, $\frac{\partial\text{FFN}(x'_j)}{\partial x'_j}$ and $\frac{\partial\text{MHSA}(x_j)}{\partial x_j}$ are also derivatives of the output of FFN and MHSA w.r.t their inputs, and can be represented as Jacobian matrices, $J_{\text{FFN}}^{x'_j}$ and $J_{\text{MHSA}}^{x_j}$, respectively. After substituting these terms, we obtain the following:

$$g_{z_i} = \frac{\partial \mathcal{L}}{\partial z_i} = P_1 \cdot \prod_{j=i+1}^{N}\left(J_{\text{LN}_2}^{z'_j} \cdot (\text{I} + J_{\text{FFN}}^{x'_j}) \cdot J_{\text{LN}_1}^{z_j} \cdot (\text{I} + J_{\text{MHSA}}^{x_j})\right) \cdot J_{\text{LN}_2}^{z'_i} \cdot (\text{I} + J_{\text{FFN}}^{x'_i}) \cdot J_{\text{LN}_1}^{z_i} \tag{54}$$

Now, we take the L2-norm on both sides of Eq. (54) as follows:

$$\|g_{z_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 = \left\|P_1 \cdot \prod_{j=i+1}^{N}\left(J_{\text{LN}_2}^{z'_j} \cdot (\text{I} + J_{\text{FFN}}^{x'_j}) \cdot J_{\text{LN}_1}^{z_j} \cdot (\text{I} + J_{\text{MHSA}}^{x_j})\right) \cdot J_{\text{LN}_2}^{z'_i} \cdot (\text{I} + J_{\text{FFN}}^{x'_i}) \cdot J_{\text{LN}_1}^{z_i}\right\|_2 \tag{55}$$

We know that the L2-norm of a matrix is equivalent to its largest singular value [Horn and Johnson, 1991]. Hence, we can further write Eq. (55) as follows:

$$\|g_{z_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 = s_{\text{max}}(P_1 \cdot \prod_{j=i+1}^{N} \Big(J_{\text{LN}_2}^{z'_j} \cdot (\text{I} + J_{\text{FFN}}^{x'_j}) \cdot J_{\text{LN}_1}^{z_j} \cdot (\text{I} + J_{\text{MHSA}}^{x_j})\Big) \cdot J_{\text{LN}_2}^{z'_i} \cdot (\text{I} + J_{\text{FFN}}^{x'_i}) \cdot J_{\text{LN}_1}^{z_i}) \tag{56}$$

where s_{max} outputs the largest singular value of $(P_1 \cdot \prod_{j=i+1}^{N} \Big(J_{\text{LN}_2}^{z'_j} \cdot (\text{I} + J_{\text{FFN}}^{x'_j}) \cdot J_{\text{LN}_1}^{z_j} \cdot (\text{I} + J_{\text{MHSA}}^{x_j})\Big) \cdot J_{\text{LN}_2}^{z'_i} \cdot (\text{I} + J_{\text{FFN}}^{x'_i}) \cdot J_{\text{LN}_1}^{z_i})$. Now from properties of singular values defined in Eq. (40), we can further rewrite Eq. (56) as follows:

$$\begin{aligned}
\|g_{z_i}\|_2 &= \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \\
&\leq s_{\text{max}}(P_1) \cdot \prod_{j=i+1}^{N} \left(s_{\text{max}}(J_{\text{LN}_2}^{z'_j}) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\big) \cdot s_{\text{max}}(J_{\text{LN}_1}^{z_j}) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{MHSA}}^{x_j})\big)\right) \\
&\qquad \cdot s_{\text{max}}(J_{\text{LN}_2}^{z'_i}) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{FFN}}^{x'_i})\big) \cdot s_{\text{max}}(J_{\text{LN}_1}^{z_i})
\end{aligned} \tag{57}$$

From Xiong et al. [2020], we can rewrite the Jacobian of LNs as follows:

$$J_{\text{LN}_1}^{z_j} = \frac{\text{I}}{\sigma_{z_j}}, \quad J_{\text{LN}_2}^{z'_j} = \frac{\text{I}}{\sigma_{z'_j}}, \quad J_{\text{LN}_2}^{z'_i} = \frac{\text{I}}{\sigma_{z'_i}}, \quad and \quad J_{\text{LN}_1}^{z_i} = \frac{\text{I}}{\sigma_{z_i}} \tag{58}$$

where σ_{z_j}, $\sigma_{z'_j}$, $\sigma_{z'_i}$ and σ_{z_i} are the standard-deviations of z_j, z'_j, z'_i, and z_i respectively. Therefore, we obtain the following equation:

$$\begin{aligned}
\|g_{z_i}\|_2 &= \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \\
&\leq s_{\text{max}}(P_1) \cdot \prod_{j=i+1}^{N} \left(s_{\text{max}}\left(\frac{\text{I}}{\sigma_{z'_j}}\right) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\big) \cdot s_{\text{max}}\left(\frac{\text{I}}{\sigma_{z_j}}\right) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{MHSA}}^{x_j})\big)\right) \\
&\qquad \cdot s_{\text{max}}\left(\frac{I}{\sigma_{z'_i}}\right) \cdot \big(s_{\text{max}}(\text{I}) + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\big) \cdot s_{\text{max}}\left(\frac{\text{I}}{\sigma_{z_i}}\right)
\end{aligned} \tag{59}$$

Another property of singular values states that all singular values of identity matrix I are 1 [Horn and Johnson, 1991], i.e., $s_k(\text{I}) = 1$. Therefore substituting with that in Eq. (59), we obtain the following:

$$\begin{aligned}
\|g_{z_i}\|_2 &= \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \\
&\leq s_{\text{max}}(P_1) \cdot \prod_{j=i+1}^{N} \left(s_{\text{max}}\left(\frac{1}{\sigma_{z'_j}}\right) \cdot \big(1 + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\big) \cdot s_{\text{max}}\left(\frac{1}{\sigma_{z_j}}\right) \cdot \big(1 + s_{\text{max}}(J_{\text{MHSA}}^{x_j})\big)\right) \\
&\qquad \cdot s_{\text{max}}\left(\frac{1}{\sigma_{z'_i}}\right) \cdot \big(1 + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\big) \cdot s_{\text{max}}\left(\frac{1}{\sigma_{z_i}}\right)
\end{aligned} \tag{60}$$

By re-arranging the terms, we finally obtain the following equation:

$$\|g_{z_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial z_i}\right\|_2 \leq s_{\text{max}}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N} \sigma_{z'_j} \sigma_{z_j}}\right) \cdot \prod_{j=i}^{N} \Big(1 + s_{\text{max}}(J_{\text{FFN}}^{x'_j})\Big) \cdot \prod_{j=i+1}^{N} \big(1 + s_{\text{max}}(J_{\text{MHSA}}^{x_j})\big) \tag{61}$$

Now, based on Eq. (49), we can re-write Eq. (61) as follows:

$$\|g_{z_i}\|_2 = \left\|\frac{\partial\mathcal{L}}{\partial z_i}\right\|_2 \leq s_{\max}(P_1)\cdot\left(\frac{1}{\prod_{j=i}^{N}\left|1-\sqrt{\text{Var}(\text{FFN}(x'_j))}\right|\left|1-\sqrt{\text{Var}(\text{MHSA}(x_j))}\right|}\right)\cdot \cdot\prod_{j=i}^{N}\left(1+s_{\max}(J_{\text{FFN}}^{x'_j})\right)\cdot\prod_{j=i+1}^{N}\left(1+s_{\max}(J_{\text{MHSA}}^{x_j})\right) \tag{62}$$

C.2 For Pre-LN model:

The Pre-LN model setup for i^{th} layer can be represented as follows:

$$\begin{aligned} x'_i &= x_i + \text{MHSA}(\text{LN}_1(x_i)) \\ y_i &= x'_i + \text{FFN}(\text{LN}_2(x'_i)) \end{aligned} \tag{63}$$

where x_i and x'_i are the inputs to LN_1 and LN_2, respectively.

Since there are two LayerNorm (LN) operations in every layer, we separately prove for both of them

C.2.1 Backpropagation analysis for LN_2 $(g_{x'_i})$:

We can write $g_{x'_i}$ for the i^{th} layer as follows:

$$g_{x'_i} = \frac{\partial\mathcal{L}}{\partial x'_i} = \frac{\partial\mathcal{L}}{\partial y_{\text{out}}}\cdot\frac{\partial y_{\text{out}}}{\partial y_N}\cdot\prod_{j=i+1}^{N}\left(\frac{\partial y_j}{\partial x'_j}\cdot\frac{\partial x'_j}{\partial x_j}\right)\cdot\frac{\partial y_i}{\partial x'_i} \tag{64}$$

Here, $\frac{\partial\mathcal{L}}{\partial y_{\text{out}}}\cdot\frac{\partial y_{\text{out}}}{\partial y_N}$, is independent of the transformers layers as they are computed using the classification head's output. Therefore we can treat them as P_2 (which does not vary across layers). Furthermore, we expand $y_j = x'_j + \text{FFN}(\text{LN}_1(x'_j))$, $x'_j = x_j + \text{MHSA}(\text{LN}_2(x_j))$, using Eq. (63), and compute their corresponding derivatives in Eq. (64). Lastly, x_{i+1} is same as y_i, because y_i is the output of the i^{th} layer which becomes input x_{i+1} of the $(i+1)^{\text{th}}$ layer.

After substituting, we get the following equation:

$$g_{x'_i} = \frac{\partial\mathcal{L}}{\partial x'_i} = P_2\cdot\prod_{j=i+1}^{N}\left(\frac{\partial(x'_j+\text{FFN}(\text{LN}_2(x'_j)))}{\partial x'_j}\cdot\frac{\partial(x_j+\text{MHSA}(\text{LN}_1(x_j)))}{\partial x_j}\right)\cdot\frac{\partial(x'_i+\text{FFN}(\text{LN}_2(x'_i)))}{\partial x'_i} \tag{65}$$

$$= P_2\cdot\prod_{j=i+1}^{N}\left((\text{I}+\frac{\partial\text{FFN}(\text{LN}_2(x'_j))}{\partial x'_j})\cdot(\text{I}+\frac{\partial\text{MHSA}(\text{LN}_1(x_j))}{\partial x_j})\right)\cdot(\text{I}+\frac{\partial\text{FFN}(\text{LN}_2(x'_i))}{\partial x'_i}) \tag{66}$$

$$= P_2\cdot\prod_{j=i+1}^{N}\left((\text{I}+\frac{\partial\text{FFN}(\text{LN}_2(x'_j))}{\partial\text{LN}_2(x'_j)}\cdot\frac{\partial\text{LN}_2(x'_j)}{\partial x'_j})\cdot(\text{I}+\frac{\partial\text{MHSA}(\text{LN}_1(x_j))}{\partial\text{LN}_1(x_j)}\cdot\frac{\partial\text{LN}_1(x_j)}{\partial x_j})\right) \cdot(\text{I}+\frac{\partial\text{FFN}(\text{LN}_2(x'_i))}{\partial\text{LN}_2(x'_i)}\cdot\frac{\partial\text{LN}_2(x'_i)}{\partial x'_i}) \tag{67}$$

Here, in Eq. (67), $\frac{\partial\text{LN}_1(x_j)}{\partial x_j}$, $\frac{\partial\text{LN}_2(x'_j)}{\partial x'_j}$, are both derivative of output of LN w.r.t their inputs, and hence can be represented as Jacobian matrices, $J_{\text{LN}_1}^{x_j}$ and $J_{\text{LN}_2}^{x'_j}$ respectively. Similarly, $\frac{\partial\text{MHSA}(\text{LN}_1(x_j))}{\partial\text{LN}_1(x_j)}$ and

$\frac{\partial \text{FFN}(\text{LN}_2(x'_j))}{\partial \text{LN}_2(x'_j)}$, are derivatives of output of MHSA/FFN w.r.t their inputs, and can also be represented as Jacobian matrices, $J_{\text{MHSA}}^{\text{LN}_1(x_j)}$ and $J_{\text{FFN}}^{\text{LN}_2(x'_j)}$ respectively.

Using these relations, we can re-write Eq. (67) as follows:

$$g_{x'_i} = \frac{\partial \mathcal{L}}{\partial x'_i} = P_2 \cdot \prod_{j=i+1}^{N} \left((\text{I} + J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j}) \cdot (\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j}) \right) \cdot (\text{I} + J_{\text{FFN}}^{\text{LN}_2(x'_i)} \cdot J_{\text{LN}_2}^{x'_i}) \tag{68}$$

We can further re-arrange the terms in Eq. (68) as follows:

$$g_{x'_i} = \frac{\partial \mathcal{L}}{\partial x'_i} = P_2 \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j} \right) \cdot \prod_{j=i+1}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right) \tag{69}$$

Now, we take the L2-norm at both sides of Eq. (69) and since we know that L2-norm of a matrix is equivalent to its maximum singular value. Therefore, we get the following equation:

$$\|g_{x'_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial x'_i} \right\|_2 = s_{\max}(P_2 \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j} \right) \cdot \prod_{j=i+1}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right)) \tag{70}$$

where $s_{\max}$ is the maximum singular value of $(P_2 \cdot \prod_{j=l}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j} \right) \cdot \prod_{j=l+1}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right))$.

From the singular values properties, discussed in Eq. (40), we can write the upper bound of $\|g_{x'_i}\|_2$ as follows:

$$\|g_{x'_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial x'_i} \right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(s_{\max}(\text{I}) + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j}) \right) \cdot \prod_{j=i+1}^{N} \left(s_{\max}(\text{I}) + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j}) \right) \tag{71}$$

We know that all singular values of an Idenitiy matrix I are 1, i.e., $s_k(\text{I}) = 1$. Thus,

$$\|g_{x'_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial x'_i} \right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j}) \right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j}) \right) \tag{72}$$

C.2.2 Backpropagation analysis for LN$_1$ (g_{x_i}):

We can write g_{x_i} for the i^{th} layer as follows:

$$g_{x_i} = \frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N} \cdot \prod_{j=i+1}^{N} \left(\frac{\partial y_j}{\partial x_{j'}} \cdot \frac{\partial x'_j}{\partial x_j} \right) \cdot \frac{\partial y_i}{\partial x_{i'}} \cdot \frac{\partial x_{i'}}{\partial x_i} \tag{73}$$

Here, $\frac{\partial \mathcal{L}}{\partial y_{\text{out}}} \cdot \frac{\partial y_{\text{out}}}{\partial y_N}$, is independent of the transformers layers as they are computed using the classification head's output. Therefore we can treat them as P_2 (which does not vary across layers). Furthermore, we expand $y_j = x'_j + \text{FFN}(\text{LN}_1(x'_j))$, $x'_j = x_j + \text{MHSA}(\text{LN}_2(x_j))$, using Eq. (63), and compute their corresponding derivatives in Eq. (73). Lastly, x_{i+1} is same as y_i, because y_i is the output of the i^{th} layer which becomes input x_{i+1} of the $(i+1)^{\text{th}}$ layer.

After substituting, we get the following equation:

$$g_{x_i} = \frac{\partial \mathcal{L}}{\partial x_i} = P_2 \cdot \prod_{j=i+1}^{N} \left(\frac{\partial (x_j' + \text{FFN}(\text{LN}_2(x_j')))}{\partial x_j'} \cdot \frac{\partial (x_j + \text{MHSA}(\text{LN}_1(x_j)))}{\partial x_j} \right) \cdot \frac{\partial (x_i' + \text{FFN}(\text{LN}_2(x_i')))}{\partial x_i'} \cdot \frac{\partial (x_i + \text{MHSA}(\text{LN}_1(x_i)))}{\partial x_i} \tag{74}$$

$$= P_2 \cdot \prod_{j=i+1}^{N} \left(\text{I} + \frac{\partial \text{FFN}(\text{LN}_2(x_j'))}{\partial x_j'} \right) \cdot \left(\text{I} + \frac{\partial \text{MHSA}(\text{LN}_1(x_j))}{\partial x_j} \right) \cdot \left(\text{I} + \frac{\partial \text{FFN}(\text{LN}_2(x_i'))}{\partial x_i'} \right) \cdot \left(\text{I} + \frac{\partial \text{MHSA}(\text{LN}_1(x_i))}{\partial x_i} \right) \tag{75}$$

$$= P_2 \cdot \prod_{j=l+1}^{N} \left((\text{I} + \frac{\partial \text{FFN}(\text{LN}_2(x_j'))}{\partial \text{LN}_2(x_j')} \frac{\partial \text{LN}_2(x_j')}{\partial x_j'}) \cdot (\text{I} + \frac{\partial \text{MHSA}(\text{LN}_1(x_j))}{\partial \text{LN}_1(x_j)} \frac{\partial \text{LN}_1(x_j)}{\partial x_j}) \right) \cdot (\text{I} + \frac{\partial \text{FFN}(\text{LN}_2(x_i'))}{\partial \text{LN}_2(x_i')} \frac{\partial \text{LN}_2(x_i')}{\partial x_i'}) \cdot (\text{I} + \frac{\partial \text{MHSA}(\text{LN}_1(x_i))}{\partial \text{LN}_1(x_i)} \frac{\partial \text{LN}_1(x_i)}{\partial x_i}) \tag{76}$$

Here, in Eq. (76), $\frac{\partial \text{LN}_1(x_j)}{\partial x_j}$, $\frac{\partial \text{LN}_2(x_j')}{\partial x_j'}$, are both derivative of output of LN w.r.t their inputs, and hence can be represented as Jacobian matrices, $J_{\text{LN}_1}^{x_j}$ and $J_{\text{LN}_2}^{x_j'}$ respectively. Similarly, $\frac{\partial \text{MHSA}(\text{LN}_1(x_j))}{\partial \text{LN}_1(x_j)}$ and $\frac{\partial \text{FFN}(\text{LN}_2(x_j'))}{\partial \text{LN}_2(x_j')}$, are derivatives of output of MHSA/FFN w.r.t their inputs, and can also be represented as Jacobian matrices, $J_{\text{MHSA}}^{\text{LN}_1(x_j)}$ and $J_{\text{FFN}}^{\text{LN}_2(x_j')}$ respectively.

Using these relations, we can re-write Eq. (76), as follows

$$g_{x_i} = \frac{\partial \mathcal{L}}{\partial x_i} = P_2 \cdot \prod_{j=i+1}^{N} \left(\left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x_j')} \cdot J_{\text{LN}_2}^{x_j'} \right) \cdot \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right) \right) \cdot \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x_i')} \cdot J_{\text{LN}_2}^{x_i'} \right) \cdot \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_i)} \cdot J_{\text{LN}_1}^{x_i} \right) \tag{77}$$

We can further re-arrange the terms in Eq. (77) as follows:

$$g_{x_i} = \frac{\partial \mathcal{L}}{\partial x_i} = P_2 \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x_j')} \cdot J_{\text{LN}_2}^{x_j'} \right) \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right) \tag{78}$$

Now, we take the L2-norm at both sides of Eq. (78) and since we know that L2-norm of a matrix is equivalent to its maximum singular value. Therefore, we get the following equation:

$$\|g_{x_i}\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial x_i} \right\|_2 = s_{\max}(P_2 \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x_j')} \cdot J_{\text{LN}_2}^{x_j'} \right) \cdot \prod_{j=i}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right)) \tag{79}$$

where $s_{\max}$ is the maximum singular value of $(P_2 \cdot \prod_{j=l}^{N} \left(\text{I} + J_{\text{FFN}}^{\text{LN}_2(x_j')} \cdot J_{\text{LN}_2}^{x_j'} \right) \cdot \prod_{j=l+1}^{N} \left(\text{I} + J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j} \right))$.

From the singular values properties, discussed in Eq. (40), we can write the upper bound of $\|g_{x_i'}\|_2$ as follows:

$$\|g_{x_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial x_i}\right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(s_{\max}(\mathrm{I}) + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j})\right) \cdot \prod_{j=i}^{N} \left(s_{\max}(\mathrm{I}) + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j})\right) \tag{80}$$

We know that all singular values of an Idenitiy matrix I are 1, i.e., $s_k(\mathrm{I}) = 1$. Thus,

$$\|g_{x_i}\|_2 = \left\|\frac{\partial \mathcal{L}}{\partial x_i}\right\|_2 \leq s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} \cdot J_{\text{LN}_2}^{x'_j})\right) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} \cdot J_{\text{LN}_1}^{x_j})\right) \tag{81}$$

□

D Theorem 3: Upper bound of the gradient norm of Early Layers LN are higher than those of Later Layers LN.

It is formally represented as follows:

$$\text{UB}(\|g_{x_1}\|_2) \geq \text{UB}(\|g_{x_2}\|_2) \geq \cdots \geq \text{UB}(\|g_{x_N}\|_2)\ ; \text{ for both Pre- and Post-LN models} \tag{82}$$

where UB($\|g_{x_i}\|_2$) denotes the upper bound of $\|g_{x_i}\|_2$ and x_i is the input to the i^{th} layer's LN.

Proof:

D.1 For Post-LN model:

D.1.1 Analysis for LN$_2$ ($g_{z'_i}$):

We prove that $\text{UB}(\|g_{z'_i}\|_2) \geq \text{UB}(\|g_{z'_{i+1}}\|_2)$, corresponding to i^{th} and $(i+1)^{\text{th}}$ layer.

We can compute $\text{UB}(\|g_{z'_i}\|_2)$ and $\text{UB}(\|g_{z'_{i+1}}\|_2)$ using Eq. (50) as follows:

$$\begin{aligned}\text{UB}(\|g_{z'_i}\|_2) =& s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N}\left|1 - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right|}\right) \cdot \left(\frac{1}{\prod_{j=i+1}^{N}\left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|}\right) \\ &\cdot \prod_{j=i+1}^{N} \left(\left(1 + s_{\max}(J_{\text{FFN}}^{x'_j})\right) \cdot \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right)\right)\end{aligned} \tag{83}$$

$$\begin{aligned}\text{UB}(\|g_{z'_{i+1}}\|_2) =& s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i+1}^{N}\left|1 - \sqrt{\text{Var}(\text{FFN}(x'_j))}\right|}\right) \cdot \left(\frac{1}{\prod_{j=i+2}^{N}\left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|}\right) \\ &\cdot \prod_{j=i+2}^{N} \left(\left(1 + s_{\max}(J_{\text{FFN}}^{x'_j})\right) \cdot \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right)\right)\end{aligned} \tag{84}$$

We then substitute these expressions in the inequality $\text{UB}(\|g_{z'_i}\|_2) \geq \text{UB}(\|g_{z'_{i+1}}\|_2)$, to prove that early layers have higher gradient norms in comparison to later layers.

$$
\begin{aligned}
&s_{\max}(P_1)\cdot\left(\frac{1}{\prod_{j=i}^{N}\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_j'))}\right|}\right)\cdot\left(\frac{1}{\prod_{j=i+1}^{N}\left|1-\sqrt{\mathrm{Var}(\mathrm{MHSA}(x_j))}\right|}\right)\\
&\quad\cdot\prod_{j=i+1}^{N}\left(\left(1+s_{\max}(J_{\mathrm{FFN}}^{x_j'})\right)\cdot\left(1+s_{\max}(J_{\mathrm{MHSA}}^{x_j})\right)\right)\\
&\quad\geq s_{\max}(P_1)\cdot\left(\frac{1}{\prod_{j=i+1}^{N}\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_j'))}\right|}\right)\cdot\left(\frac{1}{\prod_{j=i+2}^{N}\left|1-\sqrt{\mathrm{Var}(\mathrm{MHSA}(x_j))}\right|}\right)\\
&\quad\cdot\prod_{j=i+2}^{N}\left(\left(1+s_{\max}(J_{\mathrm{FFN}}^{x_j'})\right)\cdot\left(1+s_{\max}(J_{\mathrm{MHSA}}^{x_j})\right)\right)
\end{aligned}
\tag{85}
$$

This can be further rewritten as follows:

$$
\frac{(1+s_{\max}(J_{\mathrm{FFN}}^{x_{i+1}'}))(1+s_{\max}(J_{\mathrm{MHSA}}^{x_{i+1}}))}{\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right|\left|1-\sqrt{\mathrm{Var}(\mathrm{MHSA}(x_{i+1}))}\right|}\geq 1 \tag{86}
$$

From Horn and Johnson [1991], we know that every singular value of a matrix A is greater than or equal to 0, i.e., $s_k(A)\geq 0, \forall k$. Hence, for every transformer layer,

$$
(1+s_{\max}(J_{\mathrm{MHSA}}^{x_j}))\geq 1 \quad \text{and} \quad (1+s_{\max}(J_{\mathrm{FFN}}^{x_j'}))\geq 1 \tag{87}
$$

Now, to prove Eq. (86) to be true, we need to prove that

$$
0<\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right|\leq 1 \quad \& \quad 0<\left|1-\sqrt{\mathrm{Var}(\mathrm{MHSA}(x_{i+1}))}\right|\leq 1 \tag{88}
$$

We do not consider the scenario where either $\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right| = 0$ or $\left|1-\sqrt{\mathrm{Var}(\mathrm{MHSA}(x_{i+1}))}\right| = 0$, because if either/both of them becomes 0 then the gradient norm would go infinity, which we do not observe in real-world models either.

We prove Eq. (88) for the FFN component as follows (MHSA component will also have a similar proof):

Firstly, $\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right|$ can be rewritten as follows:

$$
\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right| = \begin{cases}1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}<1\\ \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}-1, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}>1\end{cases} \tag{89}
$$

We then apply the inequality described in Eq. (88) and Eq. (89) together as follows:

$$
0<\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right|\leq 1 \Rightarrow \begin{cases}0<1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\leq 1, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}<1\\ 0<\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}-1\leq 1, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}>1\end{cases} \tag{90}
$$

Further solving Eq. (90), we obtain the following:

$$
0<\left|1-\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\right|\leq 1 \Rightarrow \begin{cases}0\leq\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}<1, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}<1\\ 1<\sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}\leq 2, & \text{if } \sqrt{\mathrm{Var}(\mathrm{FFN}(x_i'))}>1\end{cases} \tag{91}
$$

Hence, the inequality $0 < \left|1 - \sqrt{\text{Var}(\text{FFN}(x_i'))}\right| \leq 1$ holds true when

$$\sigma_{\text{FFN}_i} = \sqrt{\text{Var}(\text{FFN}(x_i'))} \in [0, 2] - \{1\} \tag{92}$$

Likewise, the inequality $0 < \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_{i+1}))}\right| \leq 1$ holds true when

$$\sigma_{\text{MHSA}_{i+1}} = \sqrt{\text{Var}(\text{MHSA}(x_{i+1}))} \in [0, 2] - \{1\} \tag{93}$$

Hence, from Eq. (92) & Eq. (93), we prove that $\frac{(1+s_{\max}(J_{\text{FFN}}^{x'_{i+1}}))(1+s_{\max}(J_{\text{MHSA}}^{x_{i+1}}))}{\left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right|\left|1-\sqrt{\text{Var}(\text{MHSA}(x_{i+1}))}\right|} \geq 1$, further proving that $\text{UB}(\|g_{z_i'}\|_2) \geq \text{UB}(\|g_{z_{i+1}'}\|_2)$.

Consequently, we prove that **the upper bound of L2-norm of gradients for Early Layers $\mathbf{LN_2}$ are higher than the one of Later Layers $\mathbf{LN_2}$ in Post-LN models**, formally represented as follows:

$$\text{UB}(\|g_{z_1'}\|_2) \geq \text{UB}(\|g_{z_2'}\|_2) \geq \cdots \geq \text{UB}(\|g_{z_N'}\|_2), \tag{94}$$

when $0 < \left|1 - \sqrt{\text{Var}(\text{FFN}(x_i'))}\right| \leq 1$ and $0 < \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_{i+1}))}\right| \leq 1$.

D.1.2 Analysis for LN_1 (g_{z_i}):

We need to prove that $\text{UB}(\|g_{z_i}\|_2) \geq \text{UB}(\|g_{z_{i+1}}\|_2)$, corresponding to i^{th} and $(i+1)^{\text{th}}$ layer.

We can compute $\text{UB}(\|g_{z_i}\|_2)$ and $\text{UB}(\|g_{z_{i+1}}\|_2)$ using Eq (62) as follows:

$$\begin{aligned} \text{UB}(\|g_{z_i}\|_2) =& s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N} \left|1 - \sqrt{\text{Var}(\text{FFN}(x_j'))}\right| \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|} \right) \cdot \\ & \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{x_j'})\right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right) \end{aligned} \tag{95}$$

$$\begin{aligned} \text{UB}(\|g_{z_{i+1}}\|_2) =& s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i+1}^{N} \left|1 - \sqrt{\text{Var}(\text{FFN}(x_j'))}\right| \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|} \right) \cdot \\ & \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{x_j'})\right) \cdot \prod_{j=i+2}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right) \end{aligned} \tag{96}$$

We then substitute these expressions in the inequality $\text{UB}(\|g_{z_i}\|_2) \geq \text{UB}(\|g_{z_{i+1}}\|_2)$, to prove that early layers have higher gradients in comparison to later layers.

$$\begin{aligned} & s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i}^{N} \left|1 - \sqrt{\text{Var}(\text{FFN}(x_j'))}\right| \cdot \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|} \right) \\ & \qquad \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{x_j'})\right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right) \\ \geq \;& s_{\max}(P_1) \cdot \left(\frac{1}{\prod_{j=i+1}^{N} \left|1 - \sqrt{\text{Var}(\text{FFN}(x_j'))}\right| \cdot \left|1 - \sqrt{\text{Var}(\text{MHSA}(x_j))}\right|} \right) \\ & \qquad \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{x_j'})\right) \cdot \prod_{j=i+2}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{x_j})\right) \end{aligned} \tag{97}$$

This can be further rewritten as follows:

$$\frac{(1+s_{\max}(J_{\text{FFN}}^{x_i'}))(1+s_{\max}(J_{\text{MHSA}}^{x_{i+1}}))}{\left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right|\left|1-\sqrt{\text{Var}(\text{MHSA}(x_i))}\right|} \geq 1 \tag{98}$$

We already know that $(1+s_{\max}(J_{\text{MHSA}}^{x_{i+1}})) \geq 1 \quad \text{and} \quad (1+s_{\max}(J_{\text{FFN}}^{x_i'})) \geq 1$ from Eq. (87).

Now, to prove Eq. (98) to be true, we need to prove that

$$0 < \left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right| \leq 1 \quad \& \quad 0 < \left|1-\sqrt{\text{Var}(\text{MHSA}(x_i))}\right| \leq 1 \tag{99}$$

We do not consider the scenario where either $\left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right| = 0$ or $\left|1-\sqrt{\text{Var}(\text{MHSA}(x_i))}\right| = 0$, because if either/both of them becomes 0 then the gradient norm would go infinity, which we do not observe in real-world models either.

From Eq. (92) & Eq. (93), we know Eq. (99) holds true under the defined conditions.

Hence, we prove $\frac{(1+s_{\max}(J_{\text{FFN}}^{x_i'}))(1+s_{\max}(J_{\text{MHSA}}^{x_{i+1}}))}{\left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right|\left|1-\sqrt{\text{Var}(\text{MHSA}(x_i))}\right|} \geq 1$, further proving that $\text{UB}(\|g_{z_i}\|_2) \geq \text{UB}(\|g_{z_{i+1}}\|_2)$. This then inductively proves that **the upper bound of L2-norm of gradients for Early Layers LN_1 are greater than the one of Later Layers LN_1 in Post-LN models**, formally represented as follows:

$$\text{UB}(\|g_{z_1}\|_2) \geq \text{UB}(\|g_{z_2}\|_2) \geq \cdots \geq \text{UB}(\|g_{z_N}\|_2), \tag{100}$$

when $0 < \left|1-\sqrt{\text{Var}(\text{FFN}(x_i'))}\right| \leq 1$ and $0 < \left|1-\sqrt{\text{Var}(\text{MHSA}(x_i))}\right| \leq 1$.

E For Pre-LN model:

E.0.1 Analysis for LN_2 ($g_{x_i'}$):

We need to prove that $\text{UB}(\|g_{x_i'}\|_2) \geq \text{UB}(\|g_{x_{i+1}'}\|_2)$, corresponding to i^{th} and $(i+1)^{\text{th}}$ layer.

We can compute $\text{UB}(\|g_{x_i'}\|_2)$ and $\text{UB}(\|g_{x_{i+1}'}\|_2)$ using Eq. (72) as follows:

$$\text{UB}(\|g_{x_i'}\|_2) = s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1+s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x_j')} J_{\text{LN}_2}^{x_j'})\right) \cdot \prod_{j=i+1}^{N} \left(1+s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{101}$$

$$\text{UB}(\|g_{x_{i+1}'}\|_2) = s_{\max}(P_2) \cdot \prod_{j=i+1}^{N} \left(1+s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x_j')} J_{\text{LN}_2}^{x_j'})\right) \cdot \prod_{j=i+2}^{N} \left(1+s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{102}$$

We then substitute these expressions in the inequality $\text{UB}(\|g_{x_i'}\|_2) \geq \text{UB}(\|g_{x_{i+1}'}\|_2)$, to prove that early layers have higher gradients in comparison to later layers.

$$\begin{aligned} & s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1+s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x_j')} J_{\text{LN}_2}^{x_j'}\right)\right) \cdot \prod_{j=i+1}^{N} \left(1+s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j}\right)\right) \\ & \geq s_{\max}(P_2) \cdot \prod_{j=i+1}^{N} \left(1+s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x_j')} J_{\text{LN}_2}^{x_j'}\right)\right) \cdot \prod_{j=i+2}^{N} \left(1+s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j}\right)\right) \end{aligned} \tag{103}$$

This can be further re-written as follows:

$$\left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_i)} J_{\text{LN}_2}^{x'_i}\right)\right) \cdot \left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_{i+1})} J_{\text{LN}_1}^{x_{i+1}}\right)\right) \geq 1 \tag{104}$$

We know that every singular value of a matrix A is greater than or equal to 0, i.e., $s_k(A) \geq 0 \quad \forall k$. Hence, for every transformer layer,

$$\left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_i)} J_{\text{LN}_2}^{x'_i}\right)\right) \geq 1 \quad \text{and} \quad \left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_i)} J_{\text{LN}_1}^{x_i}\right)\right) \geq 1 \tag{105}$$

This proves that Eq. (104) is true, hence also proving that $\text{UB}(\|g_{x'_i}\|_2) \geq \text{UB}(\|g_{x'_{i+1}}\|_2) \quad \forall i$.

This then inductively proves that **the upper bound of L2-norm of gradients for Early Layers LN_2 are greater than the one of Later Layers LN_2 in Pre-LN models**, formally represented as follows:

$$\text{UB}(\|g_{x'_1}\|_2) \geq \text{UB}(\|g_{x'_2}\|_2) \geq \cdots \geq \text{UB}(\|g_{x'_N}\|_2) \tag{106}$$

E.0.2 Analysis for LN_1 (g_{x_i}):

We need to prove that $\text{UB}(\|g_{x_i}\|_2) \geq \text{UB}(\|g_{x_{i+1}}\|_2)$, corresponding to i^{th} and $(i+1)^{\text{th}}$ layer.

We can compute $\text{UB}(\|g_{x_i}\|_2)$ and $\text{UB}(\|g_{x_{i+1}}\|_2)$ using Eq. (81) as follows:

$$\text{UB}(\|g_{x_i}\|_2) = s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} J_{\text{LN}_2}^{x'_j})\right) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{107}$$

$$\text{UB}(\|g_{x_{i+1}}\|_2) = s_{\max}(P_2) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{FFN}}^{\text{LN}_2(x'_j)} J_{\text{LN}_2}^{x'_j})\right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j})\right) \tag{108}$$

We then substitute these expressions in the inequality $\text{UB}(\|g_{x_i}\|_2) \geq \text{UB}(\|g_{x_{i+1}}\|_2)$, to prove that early layers have higher gradients in comparison to later layers.

$$\begin{aligned} & s_{\max}(P_2) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_j)} J_{\text{LN}_2}^{x'_j}\right)\right) \cdot \prod_{j=i}^{N} \left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j}\right)\right) \\ & \geq s_{\max}(P_2) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_j)} J_{\text{LN}_2}^{x'_j}\right)\right) \cdot \prod_{j=i+1}^{N} \left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_j)} J_{\text{LN}_1}^{x_j}\right)\right) \end{aligned} \tag{109}$$

This can be further re-written as follows:

$$\left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_i)} J_{\text{LN}_2}^{x'_i}\right)\right) \cdot \left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_i)} J_{\text{LN}_1}^{x_i}\right)\right) \geq 1 \tag{110}$$

We know that $\left(1 + s_{\max}\left(J_{\text{FFN}}^{\text{LN}_2(x'_i)} J_{\text{LN}_2}^{x'_i}\right)\right) \geq 1$ and $\left(1 + s_{\max}\left(J_{\text{MHSA}}^{\text{LN}_1(x_i)} J_{\text{LN}_1}^{x_i}\right)\right) \geq 1$ from Eq. (105).

This proves that Eq. (110) is true, hence also proving that $\text{UB}(\|g_{x_i}\|_2) \geq \text{UB}(\|g_{x_{i+1}}\|_2) \quad \forall i$.

This then consequently proves that **the upper bound of L2-norm of gradients for Early Layers LN_1 are greater than the one of Later Layers LN_1 in Pre-LN models**, formally represented as follows:

$$\text{UB}(\|g_{x_1}\|_2) \geq \text{UB}(\|g_{x_2}\|_2) \geq \cdots \geq \text{UB}(\|g_{x_N}\|_2) \tag{111}$$

□

F Training Details

This section outlines the detailed configurations of the datasets and models used in our experiments, including dataset splits, pre-processing steps, model architectures, and training settings.

F.1 Datasets

Below we discuss the 6 datasets used in our paper.

Emotions dataset, proposed in Saravia et al. [2018], consists of 16,000 train, 2,000 validation and 2,000 test samples, each sample belonging to one of the 6 classes (class 0-5): sadness, joy, love, anger, fear and surprise. To induce the notion of noisy labels, we randomly flip labels of 1% of class 5 train samples to any another random class label and let the model train till we reach 100% train accuracy (memorizing the noisy labels).

TweetTopic dataset, developed in Antypas et al. [2022], consists of 2,858 train, 352 validation, and 376 test samples, spanning across 6 classes (class 0-5): arts_&_culture, business_&_entrepreneurs, pop_culture, science_&_technology, sports_&_gaming, daily_life. To introduce memorization of noisy labels, we flip labels of 1% of class 3 train samples to any another random class label while training the model till it reaches 100% train accuracy.

News dataset, proposed in Okite97 [2024], consists of 4,686 train and 828 test samples, spanning across 6 classes (class 0-5): business, sports, politics, health, entertainment and tech. We then split the train set to train & validation using 90:10 stratified split over the class labels. Also, we introduce noisy labels, by flipping labels of 1% of class 5 train samples to any another random class label, while letting the model overfit to 100% train accuracy.

CIFAR10 dataset, first introduced in Krizhevsky et al. [2009], consists of 60,000 samples, with each belonging to one of the 10 classes (class 0-9): airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. In our setup we consider a subset of - 16,000 train, 4,000 validation and 10,000 test samples. We also resize the images to size (224x224x3) for training the transformer model. To introduce noisy labels, we flip labels of 1% of class 9 train samples to any another random class label, and train the model till it reaches 100% train accuracy.

UTK-Face dataset, proposed in Zhang et al. [2017], consists of 23,705 samples and 5 classes (classs 0-4) depicting ethnicity: white, black, asian, indian and others. We split the dataset into train, validation and testing using 65:15:20 stratified split. We also resize the images to size (224x224x3) for training the transformer model. We then introduce noisy labels, by flipping labels of 1% of class 2 train samples to any another random class label, and train the model till it achieves 100% train accuracy.

NICO++ dataset [Zhang et al., 2023] consists of approximately 60,000 images distributed across 60 categories of everyday objects. In this study, we select a subset of 15 object categories, including car, flower, penguin, camel, chair, monitor, truck, wheat, sword, seal, lion, fish, dolphin, lifeboat, and tank. The dataset is partitioned into training (80%), validation (10%), and testing (10%) sets using stratified sampling. Images are resized to (224x224x3) for consistency with the model input requirements. We then introduce noisy labels, by flipping labels of 1% of class 6 train samples to any another random class label, and train the model till it achieves 100% train accuracy.

F.2 Models

Below we discuss the 13 transformer models (6 Post-LN and 7 Pre-LN) considered as part of our paper. We utilize the Sequence Classification variant of all the models available on Huggingface[3].

[3] `https://huggingface.co/docs/transformers/index`

Post-LN Models	Description
BERT [Devlin et al., 2019]	12-layer bidirectional transformer for masked language modeling and next sentence prediction.
DeBERTa [He et al., 2020]	12-layer model with disentangled position/content embeddings and decoding-enhanced attention.
RoBERTa [Yinhan et al., 2019]	12-layer robustly optimized BERT variant trained with dynamic masking and more data.
ELECTRA [Clark, 2020]	12-layer model using replaced token detection for sample-efficient pre-training.
DistillBERT [Sanh et al., 2019]	6-layer distilled BERT that is smaller and faster variant of BERT, retaining strong performance.
Longformer [Beltagy et al., 2020]	12-layer model with sparse attention for efficient long-sequence processing.
Pre-LN Models	**Description**
GPT2-Medium [Radford et al., 2019]	24-layer unidirectional transformer trained for causal language modeling.
GPTNeo-125M [Black et al., 2022]	12-layer open-source causal language model trained on The Pile dataset
Qwen2-0.5B [Yang et al., 2024]	24-layer efficient LLM optimized for generative tasks, using RMSNorm [Zhang and Sennrich, 2019]
RoBERTA-PreLayerNorm [Ott et al., 2019]	24-layer variant of RoBERTa with Pre-LN setting for improved training stability.
ViT-B [Alexey, 2020]	12-layer Vision Transformer Base model for image classification.
ViT-S [Assran et al., 2022]	12-layer smaller ViT variant trained with Masked Siamese Networks (MSN).
DeiT [Touvron et al., 2021]	12-layer Data-efficient Image Transformer trained with distillation, without external data.

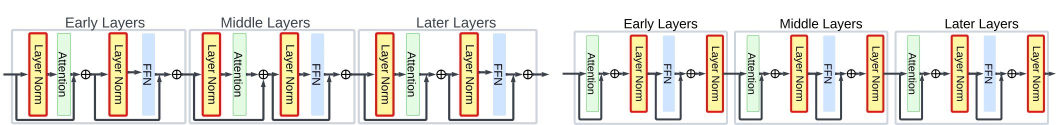
Table 3: Overview of the 13 transformer models categorized into 6 Post-LN and 7 Pre-LN architectures.

F.3 Training Settings & Hyperparameters

In our study, we explore various combinations of different models and datasets across our experiments, as follows - (1) Emotions dataset with BERT (Post-LN), DeBERTa (Post-LN), DistillBERT (Post-LN), GPTNeo (Pre-LN), (2) News dataset with ELECTRA (Post-LN), Longformer (Post-LN), Qwen2 (Pre-LN), (3) Tweets dataset with RoBERTa (Post-LN), GPT2 (Pre-LN), RoBERTa-PreLayerNorm (Pre-LN), (4) CIFAR10 dataset with ViT-B (Pre-LN), (5) UTK-Face dataset with DeiT (Pre-LN), and (6) NICO++ dataset with ViT-S (Pre-LN). We fully unfreeze all layers of the pre-trained models during training.

Regarding the hyperparameters, we consider a learning rate of 2e-5 and a batch size of 16 for all models. Then, we train the Post-LN models for 40 epochs and Pre-LN models for 70 epochs. We run 70 epochs for Pre-LN models, because after removal of LN parameters, the learning accuracy is impacted significantly in Pre-LN models from the start. Hence, we examine if the accuracy would increase by letting the model train more. In addition to that, we do not use any data augmentation in our training procedures. We also provide the code for our experiments in the supplementary file.

F.4 Grouping of Early, Middle, and Later Layers



(a) Pre-LN architecture

(b) Post-LN architecture

Figure 5: Pre-LN vs. Post-LN architectures depicting LN placement and categorization of early, middle and later layers

To investigate which group of layers' Layer Normalization (LN) contributes most to memorization and learning, we divide the transformer layers into three subsets: **Early**, **Middle**, and **Later** layers as shown in Fig. 5.

Formally, for a transformer model with N layers (where N is divisible by 3), we define:

$$\begin{aligned} \text{Early Layers} &= \{1, 2, \ldots, \tfrac{N}{3}\} \\ \text{Middle Layers} &= \left\{\tfrac{N}{3} + 1, \ldots, \tfrac{2N}{3}\right\} \\ \text{Later Layers} &= \left\{\tfrac{2N}{3} + 1, \ldots, N\right\} \end{aligned} \tag{112}$$

This grouping helps separately examine which set of layers most significantly influences learning and memorization in transformers.

G Additional Experiments & Results

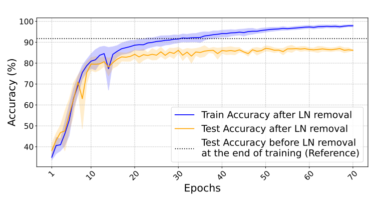
This section presents additional experiments (on top of the ones discussed in Sec. 4, Sec. 5 and Sec. 6)and results on supplementary datasets and models, expanding on the analyses in Sec. G.1, G.2, and G.3. These experiments aim to: (1) establish the distinctive impact of Layer Normalization (LN) on memorization and learning in Pre-LN vs. Post-LN models, (2) assess the role of LN in early layers, and (3) investigate how gradient behavior accounts for the observed phenomena.

G.1 Impact of LN on Memorization & Learning in Pre- and Post-LN models

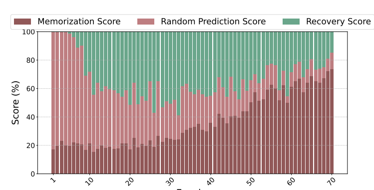
In this section, we present the results corresponding to the distinctive impact of LN of memorization and learning for the remaining Pre and Post LN models spanning multiple datasets. These results further corroborate our finding that removal of LN parameters in pre-LN models critically destabilizes learning while exacerbating overfitting, while for Post-LN models, LN parameters removal, suppresses memorization and facilitates true label recovery without impacting learning.

The results are verified against Pre-LN models - GPTNeo, GPT2, ViT-B, DeiT, ViT-S, RoBERTa-PreLayernorm and Post-LN models - BERT, DeBERTa, Longformer, RoBERTa, DistilBERT spanning across multiple language and vision datasets - Emotions, News, Tweets, CIFAR10, NICO++, UTK-Face, are provided in Figs. 6, 7, 8, 9, 10, 11, 12, 13, 14,15. and 16.

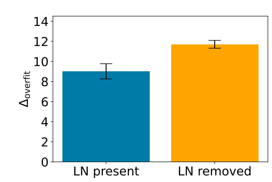
G.1.1 Pre-LN models - Learning Destabilized



(a) Learning (Test) accuracy over epochs for Pre-LN Model (GPTNeo)

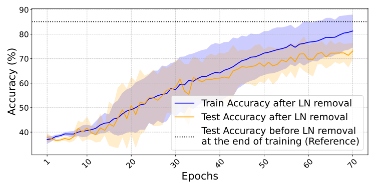


(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (GPTNeo)

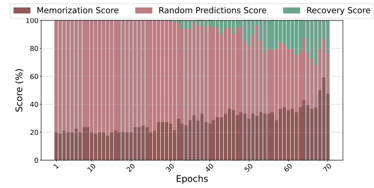


(c) Overfitting gap for Pre-LN Model (GPTNeo)

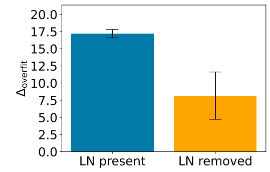
Figure 6: **LN removal destabilizes learning in Pre-LN model - GPTNeo, Emotions Dataset:** LN removal critically affects learning while memorization still persists in GPTNeo. This further exacerbates overfitting, explained by increasing train-test accuracy gap when LN is removed, due to the drop in test-accuracy.



(a) Learning (Test) accuracy over epochs for Pre-LN Model (GPT2)

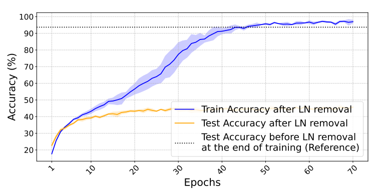


(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (GPT2)

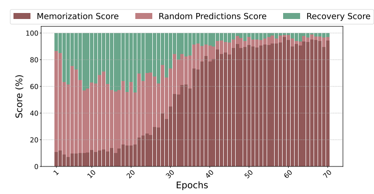


(c) Overfitting gap for Pre-LN Model (GPT2)

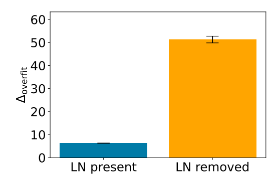
Figure 7: **LN removal destabilizes learning in Pre-LN model - GPT2, TweetTopic Dataset:** LN removal critically affects learning while memorization still persists in GPT2. For GPT2, the overfitting gap decreased after LN removal, because the model could not even stabilize during training due to the destabilization of learning. Hence both train and test accuracies remain low and comparable. However, the learning accuracy still remains low when LN is absent in comparison to when LN was present, and struggle with high memorization and random predictions (red-color family bars).



(a) Learning (Test) accuracy over epochs for Pre-LN Model (ViT-B)

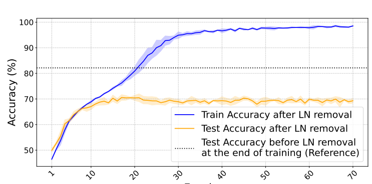


(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (ViT-B)

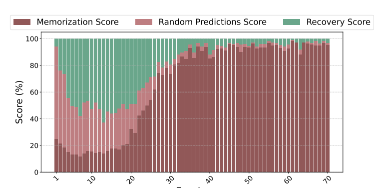


(c) Overfitting gap for Pre-LN Model (ViT-B)

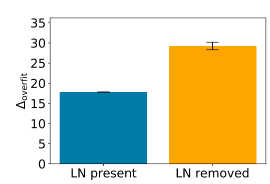
Figure 8: **LN removal destabilizes learning in Pre-LN model - ViT-B, CIFAR10 Dataset:** LN removal critically affects learning while memorization still persists in ViT-B. This further exacerbates overfitting seen by increasing train-test accuracy gap when LN is removed, due to the drop in test-accuracy.



(a) Learning (Test) accuracy over epochs for Pre-LN Model (DeiT)

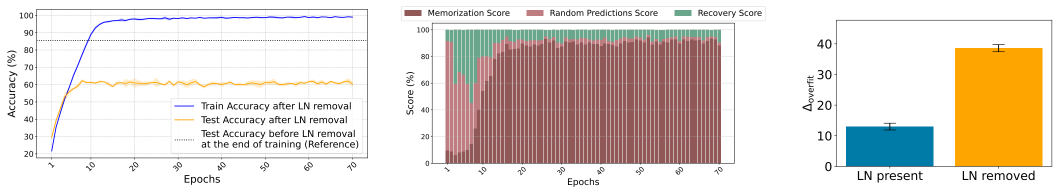


(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (DeiT)



(c) Overfitting gap for Pre-LN Model (DeiT)

Figure 9: **LN removal destabilizes learning in Pre-LN model - DeiT, UTK-Face Dataset:** LN removal critically affects learning while memorization still persists in DeiT. This further exacerbates overfitting seen by increasing train-test accuracy gap when LN is removed, due to the drop in test-accuracy.

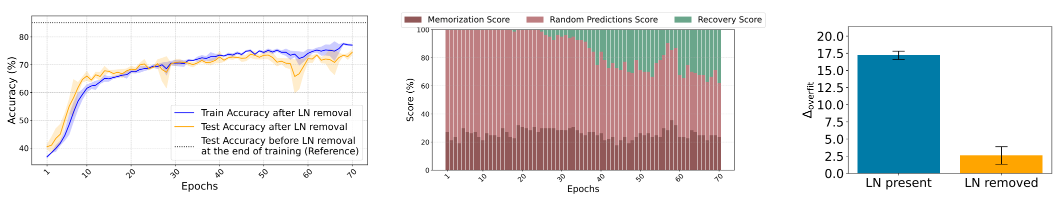


(a) Learning (Test) accuracy over epochs for Pre-LN Model (ViT-S)

(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (ViT-S)

(c) Overfitting gap for Pre-LN Model (ViT-S)

Figure 10: **LN removal destabilizes learning in Pre-LN model - ViT-S, NICO++ Dataset:** LN removal critically affects learning while memorization still persists in ViT-S. This further exacerbates overfitting seen by increasing train-test accuracy gap when LN is removed, due to the drop in test-accuracy.



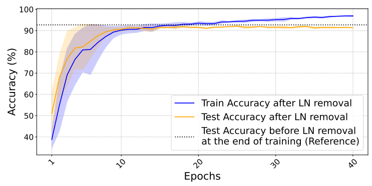
(a) Learning (Test) accuracy over epochs for Pre-LN Model (RoBERTa-PreLayerNorm)

(b) Memorization, Recovery and Random Predictions over epochs for Pre-LN Model (RoBERTa-PreLayerNorm)

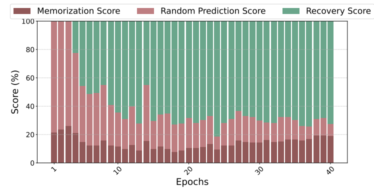
(c) Overfitting gap for Pre-LN Model (RoBERTa-PreLayerNorm)

Figure 11: **LN removal destabilizes learning in Pre-LN model - RoBERTa-PreLayerNorm, TweetTopic Dataset:** LN removal critically affects learning while memorization still persists in RoBERTa-PreLayerNorm. For RoBERTa-PreLayerNorm, the overfitting gap decreased after LN removal, because the model could not even stabilize during training due to the destabilization of learning. Hence both train and test accuracies remain low and comparable. However, the learning accuracy still remains low when LN is absent in comparison to when LN is present, and struggles with high memorization and random predictions (red-color family bars).

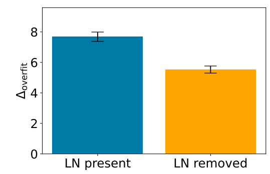
G.1.2 Post-LN models - Suppression of Memorization & True Labels Recovery



(a) Learning (Test) accuracy over epochs for Post-LN Model (BERT)

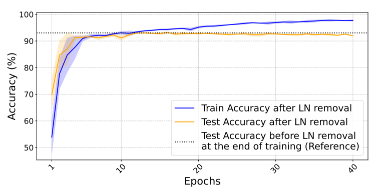


(b) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (BERT)

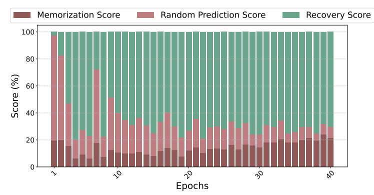


(c) Overfitting gap for Post-LN Model (BERT)

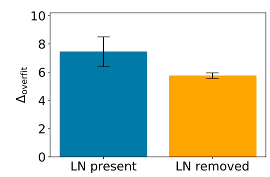
Figure 12: **LN removal suppresses memorization & facilitates true label recovery in Post-LN model - BERT, Emotions Dataset:** LN removal in BERT suppresses memorization and facilitates true label recovery, while keeping learning intact. This further reduces overfitting seen by decreasing train-test accuracy gap when LN is removed.



(a) Learning (Test) accuracy over epochs for Post-LN Model (DeBERTa)

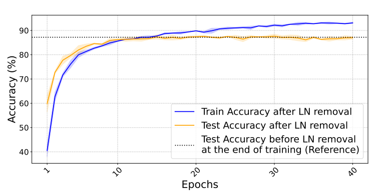


(b) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (DeBERTa)

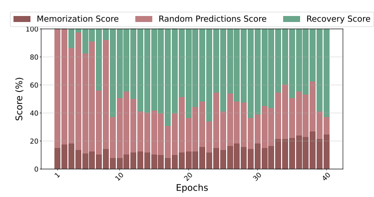


(c) Overfitting gap for Post-LN Model (DeBERTa)

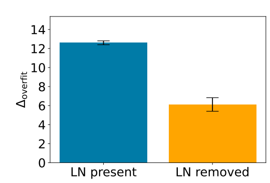
Figure 13: **LN removal suppresses memorization & facilitates true label recovery in Post-LN model - DeBERTa, Emotions Dataset:** LN removal in DeBERTa suppresses memorization and facilitates true label recovery, while keeping learning intact. This further reduces overfitting seen by decreasing train-test accuracy gap when LN is removed.



(a) Learning (Test) accuracy over epochs for Post-LN Model (Longformer)

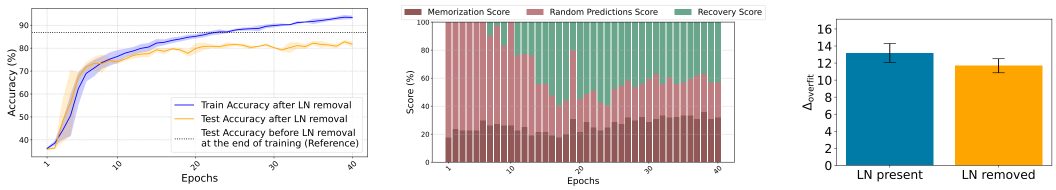


(b) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (Longformer)



(c) Overfitting gap for Post-LN Model (Longformer)

Figure 14: **LN removal suppresses memorization & facilitates true label recovery in Post-LN model - Longformer, News Dataset:** LN removal in Longformer suppresses memorization and facilitates true label recovery, while keeping learning intact. This further reduces overfitting seen by decreasing train-test accuracy gap when LN is removed.

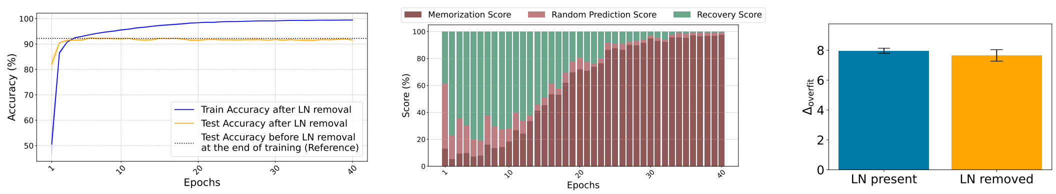


(a) Learning (Test) accuracy over epochs for Post-LN Model (RoBERTa)

(b) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (RoBERTa)

(c) Overfitting gap for Post-LN Model (RoBERTa)

Figure 15: **LN removal suppresses memorization & facilitates true label recovery in Post-LN model - RoBERTa, TweetTopic Dataset:** LN removal in Longformer suppresses memorization and facilitates true label recovery, while minimal drop in learning. This further reduces overfitting seen by decreasing train-test accuracy gap when LN is removed. *Note: For RoBERTa we see a slight drop in learning accuracy which is much lower in comparison to the huge drop which happens in GPT2 setup in Fig. 7. Furthermore, memorization is still suppressed in comparison to GPT2 where memorization still persisted.*



(a) Learning (Test) accuracy over epochs for Post-LN Model (DistilBERT)

(b) Memorization, Recovery and Random Predictions over epochs for Post-LN Model (DistilBERT)

(c) Overfitting gap for Post-LN Model (DistilBERT)

Figure 16: **LN removal does not suppress memorization in DistilBERT (Emotions Dataset) but does not affect learning:** LN removal in DistilBERT does not suppress memorization (but it does not impact learning) as we observe for all other 5 Post-LN models. We think other components in the Transformer architecture might have a more profound impact on memorization in DistilBERT. But, since this work is primarily about LN impact, we refrain from dealing with other components, making it an interesting future work.

G.2 Significance of Early Layers LN

In this section, we illustrate the results corresponding to the significance of Early Layers LN in impacting learning and suppressing memorization for remaining Pre- and Post-LN models, respectively, across multiple datasets.

The results verified against Pre-LN models - GPTNeo, Qwen2, GPT2, RoBERTa-PreLayernorm, ViT-B, ViT-S, and Post-LN models - BERT, ELECTRA, Longformer, RoBERTa, DistilBERT are shown in Figs. 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, and 27.

G.2.1 Pre-LN Models: Early Layers LN drives learning

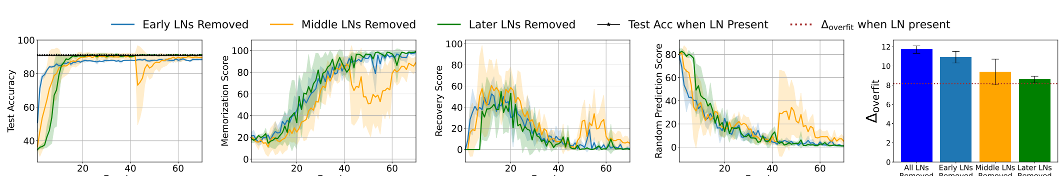


Figure 17: **Pivotal impact of early LNs for learning in Pre-LN model (GPTNeo, Emotions Dataset).** We can clearly observe the impact of early layers LN removal on destabilizing learning in GPTNeo, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than for later layers, $\Delta_{\text{overfit}}^{\text{Pre, later}}$, and poor memorization suppression.

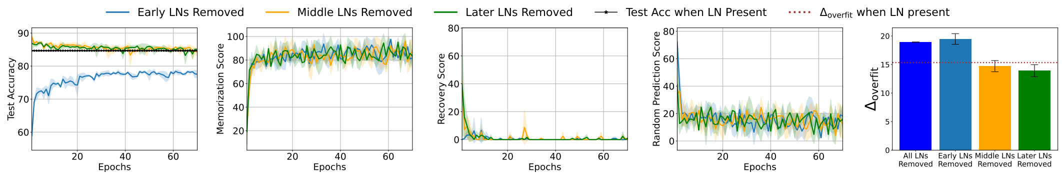


Figure 18: **Pivotal impact of early LNs for learning in Pre-LN model (Qwen2, News Dataset).** We can clearly observe the impact of early layers LN removal on destabilizing learning in Qwen2, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than for later layers, $\Delta_{\text{overfit}}^{\text{Pre, later}}$, and poor memorization suppression.

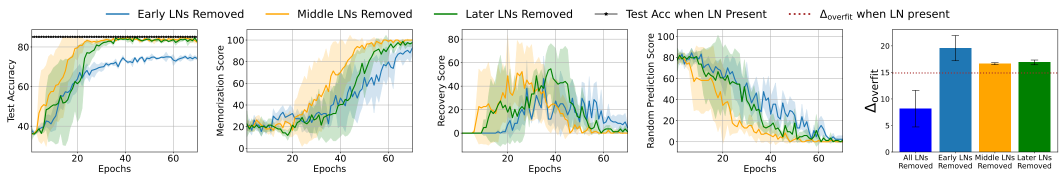


Figure 19: **Pivotal impact of early LNs for learning in Pre-LN model (GPT2, TweetTopic Dataset).** We can clearly observe the impact of early layers LN removal on destabilizing learning in GPT2, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than for later layers, $\Delta_{\text{overfit}}^{\text{Pre, later}}$, and poor memorization suppression. *Note: When all layers LNs are removed, $\Delta_{overfit}^{Pre, all}$ is low because when we removed all LNs from GPT2 then the model could not stabilize in training, reaching very low train accuracy, similar train-test accuracy as seen in Fig. 7a, hence low $\Delta_{overfit}^{Pre, all}$. However, when we just removed early LNs, the model is able to converge to a sufficiently high train accuracy, but the learning is impacted much severely in comparison to later layers.*

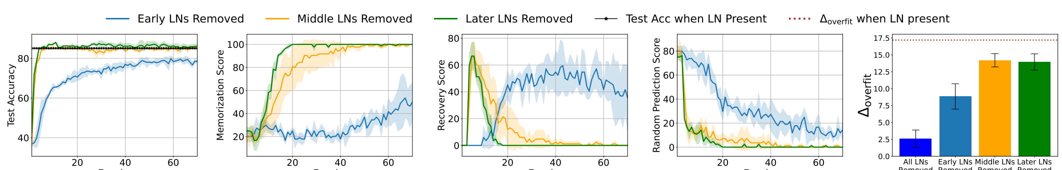


Figure 20: **Pivotal impact of early LNs for learning in Pre-LN model (RoBERTa-PreLayerNorm, TweetTopic Dataset).** We observe that for RoBERTa-PreLayerNorm, removing early layers LN impacts learning, however, memorization also seems to get suppressed, when compared with removing all LNs where we observe persistent memorization and destabilized learning (Fig. 11a). Despite this unique trend of RoBERTa-PreLayernorm (not following the consistent trend just as in the other 6 Pre-LN models), early layers LN still remain the most significant.

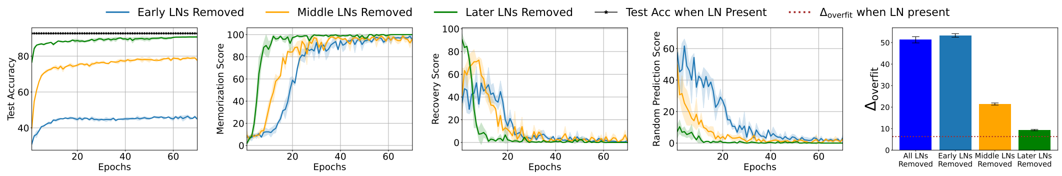


Figure 21: **Pivotal impact of early LNs for learning in Pre-LN model (ViT-B, CIFAR10 Dataset).** We can clearly observe the impact of early layers LN removal on destabilizing learning in ViT-B, accompanied with higher train-test-accuracy gap, $\Delta^{\text{Pre, early}}_{\text{overfit}}$, than for later layers, $\Delta^{\text{Pre, later}}_{\text{overfit}}$, and poor memorization suppression.

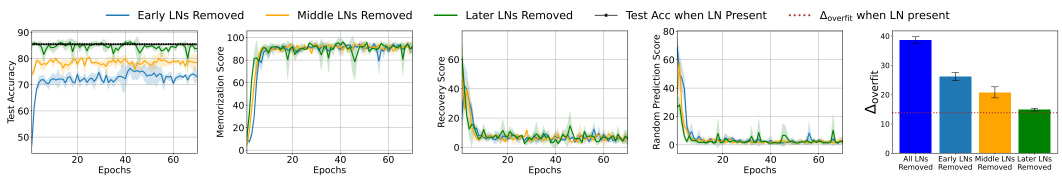


Figure 22: **Pivotal impact of early LNs for learning in Pre-LN model (CIFAR100, ViT-S Dataset).** We can clearly observe the impact of early layers LN removal on destabilizing learning in ViT-S, accompanied with higher train-test-accuracy gap, $\Delta^{\text{Pre, early}}_{\text{overfit}}$, than for later layers, $\Delta^{\text{Pre, later}}_{\text{overfit}}$, and poor memorization suppression.

G.2.2 Post-LN Models: Early Layers LN suppresses memorization

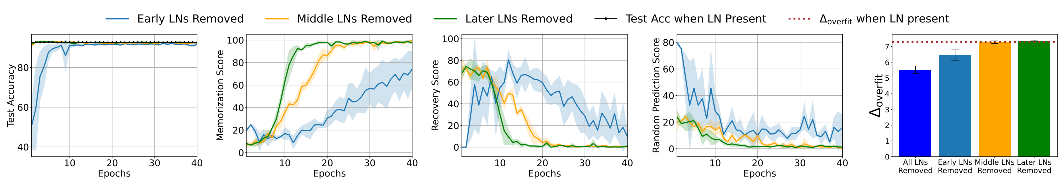


Figure 23: **Pivotal impact of early LNs on memorization in Post-LN model (BERT, Emotions Dataset).** We can clearly observe the impact of early layers LN removal on suppressing memorization & achieving true label recovery in BERT, accompanied with higher train-test-accuracy gap, $\Delta^{\text{Pre, early}}_{\text{overfit}}$, than for later layers, $\Delta^{\text{Pre, later}}_{\text{overfit}}$, while learning being intact.

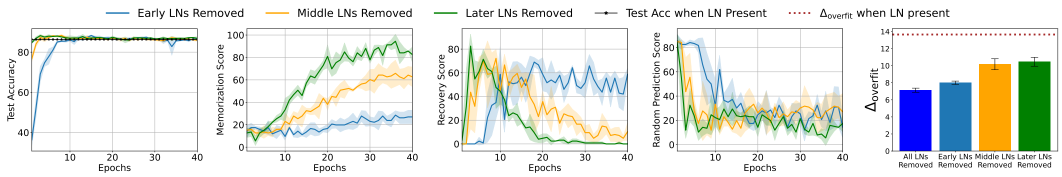


Figure 24: **Pivotal impact of early LNs on memorization in Post-LN model (ELECTRA, News Dataset).** We can clearly observe the impact of early layers LN removal on suppressing memorization & achieving true label recovery in ELECTRA, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than for later layers, $\Delta_{\text{overfit}}^{\text{Pre, later}}$, while learning being intact.

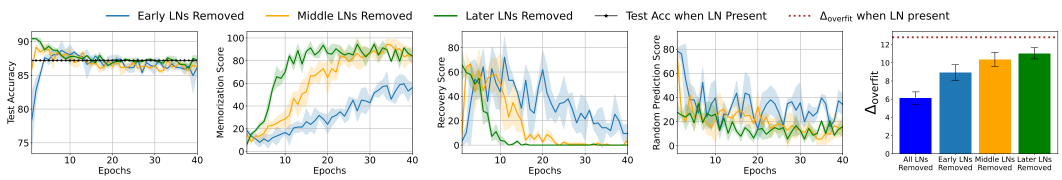


Figure 25: **Pivotal impact of early LNs on memorization in Post-LN model (Longformer, News Dataset).** We can clearly observe the impact of early layers LN removal on suppressing memorization & achieving true label recovery in Longformer, accompanied with higher train-test-accuracy gap, $\Delta_{\text{overfit}}^{\text{Pre, early}}$, than for later layers, $\Delta_{\text{overfit}}^{\text{Pre, later}}$, while learning being intact.

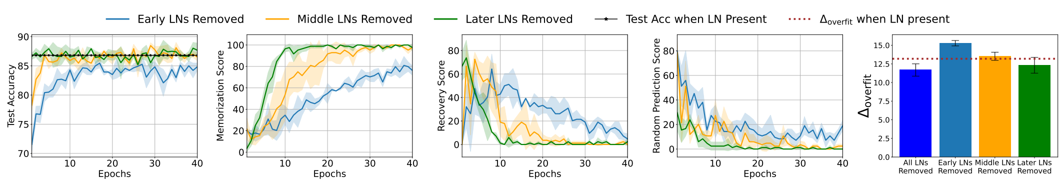


Figure 26: **Pivotal impact of early LNs on memorization in Post-LN model (RoBERTa, Tweet-Topic Dataset).** We can clearly observe the impact of early layers LN removal on suppressing memorization & achieving true label recovery in RoBERTa, while learning being minimally impacted. *Note: In the case of RoBERTa, $\Delta_{overfit}^{Pre, early}$ is slightly higher than $\Delta_{overfit}^{Pre, early}$, because even though removing early LNs, led to a greater memorization suppression than later LNs, but learning also got affected slightly, hence, the overfitting gap increased. However, we need to compare this Post-LN result with its Pre-LN counterpart of GPT2 (Fig. 19), where early LNs removal, affected learning even more, without suppressing memorization.*

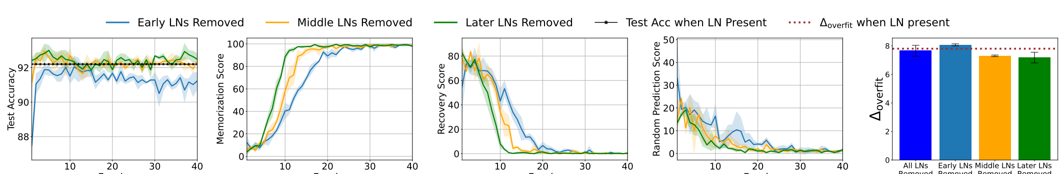


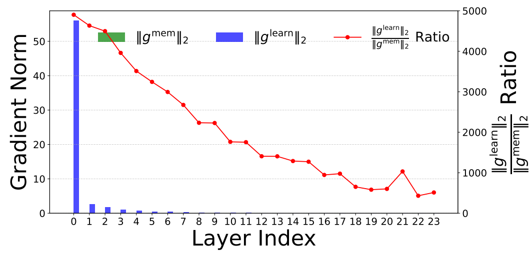
Figure 27: **Pivotal impact of early LNs on memorization in Post-LN model (DistilBERT, Emotions Dataset).** For the case of DistilBERT, we observe that just like when all LNs removal could not suppress memorization, the same happens with early LNs removal. However, we want to draw attention to the delay in achieving memorization, i.e., early LNs removal results in the highest delay in memorization compared to Middle/Later LNs removal. This shows that Early LNs are still the most significant.

G.3 Gradients explain the impact of LN on Memorization & Learning

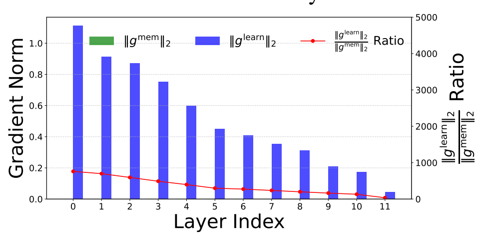
In this section, we provide additional results to understand the distinctive impact of LN on memorization and learning through the lens of gradients. These results provide a deeper understanding of why (1) LN removal destabilizes learning in Pre-LN models, and suppresses memorization in Post-LN models, (2) Early Layers LN are more significant than later layers LN in driving these phenomena.

G.3.1 Language Datasets Results

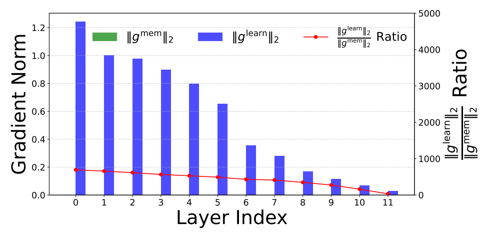
In the language modality, we experimented with additional Pre-LN (GPTNeo, GPT2, Qwen2) and Post-LN models (BERT, RoBERTa, Longformer, ELECTRA) across Emotions, News, and Tweets Datasets. The results for the same are depicted in Figs. 28, 29, and 30.



(a) Pre-LN (Qwen2) $\left\|g_x^{\text{learn}}\right\|_2$ & $\left\|g_x^{\text{mem}}\right\|_2$ analysis across layers

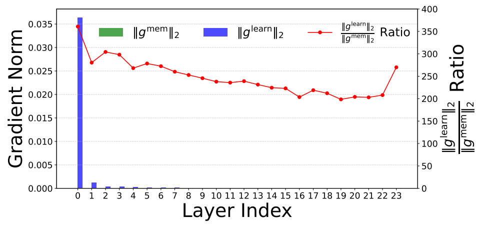


(b) Post-LN (ELECTRA) $\left\|g_x^{\text{learn}}\right\|_2$ & $\left\|g_x^{\text{mem}}\right\|_2$ analysis across layers

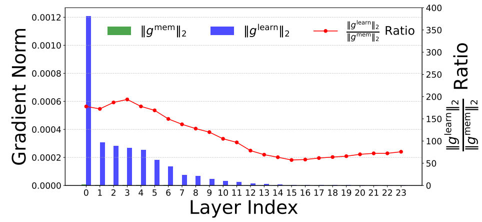


(c) Post-LN (Longformer) $\left\|g_x^{\text{learn}}\right\|_2$ & $\left\|g_x^{\text{mem}}\right\|_2$ analysis across layers

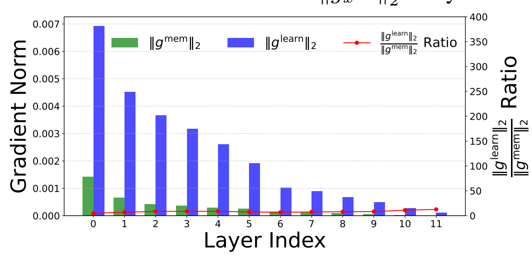
Figure 28: **Learning vs. Memorization Gradients in Pre- and Post-LN Models (News Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (Qwen2) and Post-LN (ELECTRA, Longformer) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\left\|g_x^{\text{mem}}\right\|_2$) across all layers. Furthermore, the ratio $\left\|g_x^{\text{learn}}\right\|_2 / \left\|g_x^{\text{mem}}\right\|_2$ is significantly higher in Pre-LN models compared to Post-LN models.



(a) Pre-LN (GPT2) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

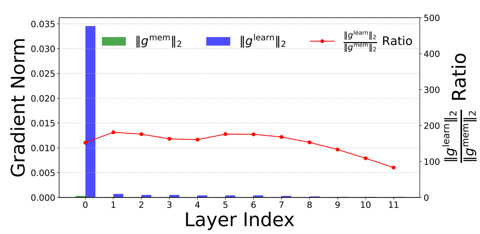


(b) Pre-LN (RoBERTa-PreLayerNorm) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

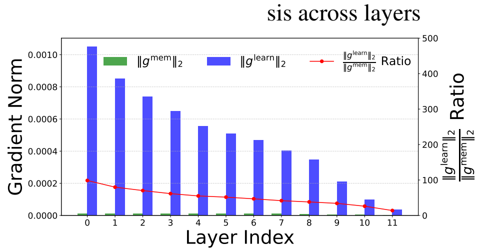


(c) Post-LN (RoBERTa) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

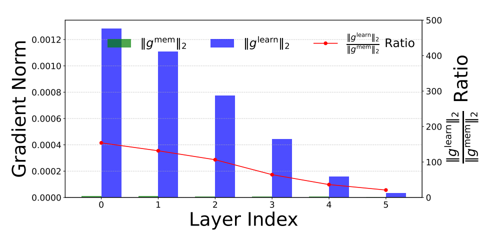
Figure 29: **Learning vs. Memorization Gradients in Pre- and Post-LN Models (TweetTopic Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (GPT2, RoBERTA-PreLayerNorm) and Post-LN (RoBERTa) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\|g_x^{\text{mem}}\|_2$) across all layers. Furthermore, the ratio $\left\|g_x^{\text{learn}}\right\|_2 / \|g_x^{\text{mem}}\|_2$ is significantly higher in Pre-LN models compared to Post-LN models.



(a) Pre-LN (GPTNeo) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers



(b) Post-LN (BERT) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

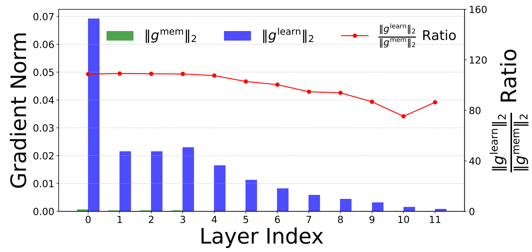


(c) Post-LN (DistilBERT) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

Figure 30: **Learning vs. Memorization Gradients in Pre- and Post-LN Models (Emotions Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (GPTNeo) and Post-LN (BERT, DistilBERT) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\|g_x^{\text{mem}}\|_2$) across all layers. Furthermore, the ratio $\left\|g_x^{\text{learn}}\right\|_2 / \|g_x^{\text{mem}}\|_2$ is significantly higher in Pre-LN models compared to Post-LN models.

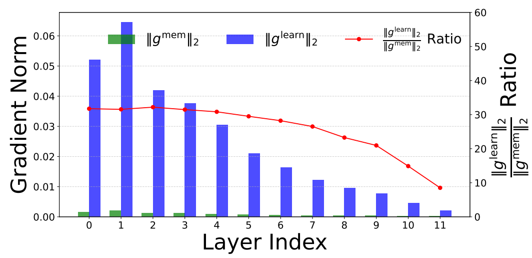
G.3.2 Vision Datasets Results

For the vision modality, it needs to be acknowledged that Post-LN architectures are not available in practice/literature, and only Pre-LN models are available. Hence, we provide additional experiments for Pre-LN models - ViT-B, ViT-S, and DeiT using multiple datasets - CIFAR10, CIFAR100, UTK-Face. The results for the same are presented in Figs. 31, 32, and 33.



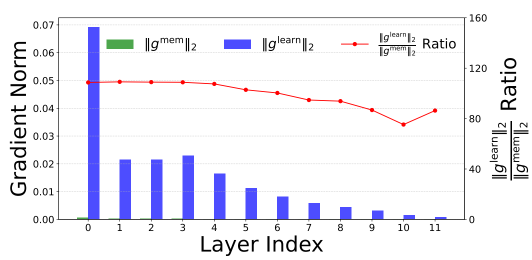
(a) Pre-LN (ViT-B) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

Figure 31: **Learning vs. Memorization Gradients in Pre-LN Models (CIFAR10 Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (ViT-B) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\|g_x^{\text{mem}}\|_2$) across all layers.



(a) Pre-LN (ViT-S) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

Figure 32: **Learning vs. Memorization Gradients in Pre-LN Models (NICO++ Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (ViT-S) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\|g_x^{\text{mem}}\|_2$) across all layers.



(a) Pre-LN (DeiT) $\left\|g_x^{\text{learn}}\right\|_2$ & $\|g_x^{\text{mem}}\|_2$ analysis across layers

Figure 33: **Learning vs. Memorization Gradients in Pre-LN Models (UTK-Face Dataset):** Results clearly exhibit high gradient norms of early layers LNs than later layers for both learning and memorization in Pre-LN (DeiT) models. Importantly, the learning gradient norms ($\left\|g_x^{\text{learn}}\right\|_2$) are consistently higher than the memorization gradient norms ($\|g_x^{\text{mem}}\|_2$) across all layers.

H Analysis across multiple Noisy Label Ratios and Optimizer

We extend our analysis to evaluate whether the claim—*removing LN parameters mitigates memorization in Post-LN models while impairing generalization in Pre-LN models*—holds consistently across higher noisy label ratios (2% and 5%) in the training dataset, as shown in Table 4.

Noise	Model	Setting	Learning (↑)	Memorization (↓)	Recovery (↑)	Random Prediction (↓)
2%	**Post-LN (BERT)**	Before	91.70	100.00	0.00	0.00
		After	92.00	20.62	76.25	3.12
	Pre-LN (GPT-Neo)	Before	91.35	100.00	0.00	0.00
		After	84.85	66.87	16.56	16.56
5%	**Post-LN (BERT)**	Before	90.35	100.00	0.00	0.00
		After	91.25	27.00	66.88	6.12
	Pre-LN (GPT-Neo)	Before	90.35	100.00	0.00	0.00
		After	82.60	67.50	11.00	21.50

Table 4: Results for higher noisy label ratios (2% and 5%) on the Emotions dataset.

Additionally, we assess the robustness of this finding under a different optimization setting by experimenting with Muon optimizer other than Adam, which was used in the primary experiments, as shown in Table 5.

Model	Setting	Learning (↑)	Memorization (↓)	Recovery (↑)	Random Prediction (↓)
Post-LN (BERT)	Before	91.95	100.00	0.00	0.00
	After	92.00	25.00	62.50	12.50
Pre-LN (GPT-Neo)	Before	91.70	100.00	0.00	0.00
	After	85.55	32.50	29.38	38.12

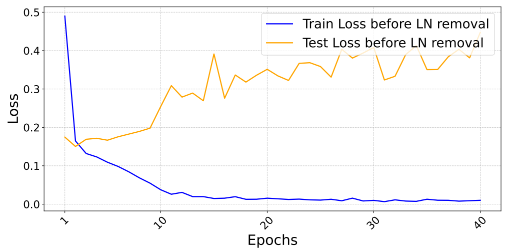
Table 5: Results with the Muon optimizer with Emotions Dataset.

Overall, the results demonstrate that our observations are agnostic to optimizer and noise-label ratios.

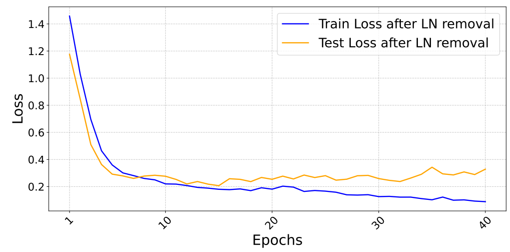
I Loss and Gradient Norms Analysis across Epochs

In Sec. 4, we showed how on LN parameters removal in Post-LN models, memorization is mitigated over epochs, while in Pre-LN models, the testing accuracy drops as training progresses, without recovering at any time. To further corroborate this claim, we plot the train-test loss functions for Post-LN and Pre-LN models, where the overfitting gap decreases in Post-LN models, while for Pre-LN models it increases, upon LN removal, as shown in Fig. 34.

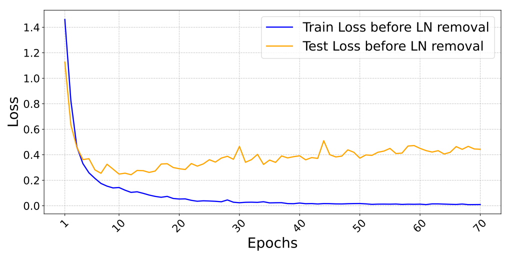
Apart from the loss curves, we also provide how the gradient norm evolves across epochs for both Post-LN and Pre-LN models in Fig. 35.



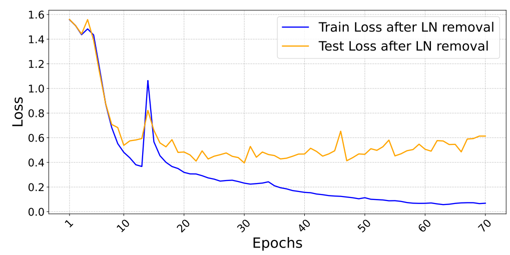
(a) BERT (Post-LN) before LN removal.



(b) BERT (Post-LN) after LN removal.

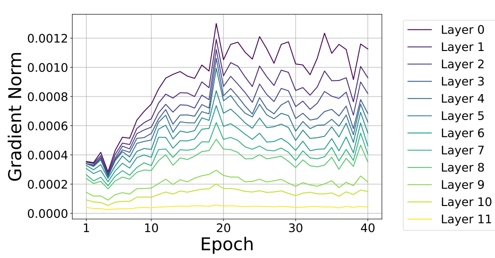


(c) GPT-Neo (Pre-LN) before LN removal.

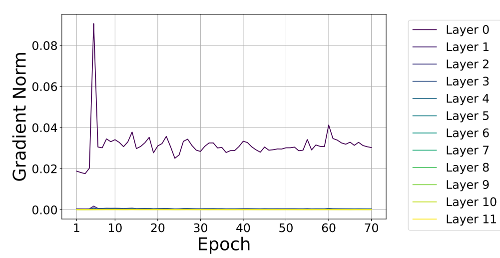


(d) GPT-Neo (Pre-LN) after LN removal.

Figure 34: Train and test loss trends across epochs for Post-LN (BERT) and Pre-LN (GPT-Neo) models before and after LayerNorm (LN) removal. In Post-LN models, LN removal reduces the overfitting gap (difference between train and test losses), mitigating memorization. In Pre-LN models, LN removal widens the overfitting gap, indicating impaired generalization.



(a) Gradient norms for BERT



(b) Gradient norms for GPT-Neo

Figure 35: Gradient norms gradually increase across epochs for both Pre-LN and Post-LN models, with early layers exhibiting the highest gradient norms.

J Validity of Results for Generative Modeling Task

To verify the robustness of our findings in a generative modeling context, we conduct a Next Token Prediction (NTP) experiment on the Emotions dataset using both BERT (Post-LN) and GPT-Neo (Pre-LN) models.

Each input sample is reformulated as:

$$\text{original text} + \text{“This emotion is [type]”}$$

where the model predicts the token corresponding to the emotion label $[type]$, one of six possible emotion types. To introduce **noisy labels**, we randomly replace the $[type]$ token for 1% of the training samples with a different emotion label. Two different model configurations are trained and evaluated: (i) before LN parameters removal, and (ii) after LN parameters removal.

As shown in Table 6, removing LN parameters substantially reduces memorization in Post-LN models (BERT), while impairing learning and generalization in Pre-LN models (GPT-Neo). This confirms that the trends reported in the main paper hold consistently for generative modeling tasks as well.

Table 6: Results on Next Token Prediction task with noisy labels.

Model	**Setting**	**Learning (↑)**	**Memorization (↓)**	**Recovery (↑)**	**Random Prediction (↓)**
Post-LN (BERT)	Before	92.14	100.00	0.00	0.00
	After	91.95	28.12	60.62	11.25
Pre-LN (GPT-Neo)	Before	91.70	100.00	0.00	0.00
	After	85.55	51.25	19.38	29.38

K Broader Impacts and Limitations

Our study reveals that LayerNorm (LN) affects memorization and learning differently across transformer variants: disabling LN parameters suppresses memorization and aids label recovery in Post-LN models, but destabilizes learning in Pre-LN models. These insights can inform architecture design and robust training in noisy labels settings, where memorization needs to be controlled. While we focus on LN due to its central role and controllability, other components—like residual paths, attention, and feedforward layers—also influence memorization and merit further investigation. We hope this work can provide insights to the community to encourage further follow-up studies.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have systematically demonstrated the distinct impact of LN on memorization and learning across Pre- and Post-LN models using various experiments and results in Sec. 4, 5, and 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, the paper reflects the limitations of prior work, while emphasizing the novel results of the distinctive impact of LN on memorization and learning.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes, complete and correct proofs are provided for the 3 Theorems provided in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, the paper fully discloses all the important information in replicating the experiments done in it.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes, code corresponding to the results is provided in the supplementary file.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, proper training details (data splits, hyper-parameters, optimizer, etc.) are provided in the Appendix. Furthermore, the code corresponding to the results is also provided in the supplementary file.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Yes, the experiments are done across 3 random seeds for robustness in results, while providing necessary error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, necessary compute resources specification are provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the paper conforms, in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, we have added a seperate section on discussing broader impacts and limitations of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, all existing datasets are properly cited in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, proper documented code is provided for the new experiments done in the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.