Understanding Normalization Layers for Sparse Training

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2025

Abstract

Normalization Layers have become an essential component of Deep Neural Networks for better training dynamics and convergence rate. While the effects of layers like BatchNorm and LayerNorm are well studied for dense networks, their impact on the training dynamics of Sparse Neural Networks (SNNs) is not well understood. In this work, we analyze the role of Batch Normalization (BN) in the training dynamics of SNNs. We theoretically and empirically show that BatchNorm induces training instability in SNNs, leading to lower convergence rates and worse generalization performance compared to the dense models. Specifically, we show that adding BatchNorm layers into sparse neural networks can significantly increase the gradient norm, causing training instability. We further validate this instability by analyzing the operator norm of the Hessian, finding it substantially larger in the case of sparse training that the dense training. This indicates that the sparse training operates further beyond the "edge of stability" bound of $2/\eta$. To mitigate this instability, we propose a novel preconditioned gradient descent method for sparse networks with BatchNorm. Our method takes into account the sparse topology of the neural network and rescales the gradients to prevent blow-up. We empirically demonstrate that our proposed preconditioned gradient descent improves the convergence rate and generalization for Dynamic Sparse Training.

1. Introduction

In recent years, Machine Learning (ML) has achieved remarkable performance across a diverse range of tasks. This remarkable success of ML could be attributed to an exponential increase in the size of the models [12], which increases the computational cost of training and deploying large foundational models. To that end, model pruning and quantization have gained significant traction to reduce the computation requirement for deploying foundational models; however, they do not reduce the computational cost associated with training. In contrast to post-training model compression methods, sparse training leverages weight sparsity during training, making it more computationally efficient. Currently, Sparse Training methods do not match dense model generalization and have slower convergence rates — taking more epochs to converge, effectively requiring the same FLOPs as dense models. In order for sparse training to be practically useful, it is therefore important to improve the training convergence rate. One potential reason for the slower convergence of sparse training methods is using the same training components (optimizers, normalization layers, activation functions) as dense models.

Batch Normalization (BN), proposed by [11], was introduced to improve training stability and convergence rate of Deep Neural Networks (DNNs). BN works by normalizing the intermediate activations of the Neural Network (NN) by centering and normalizing with the batch mean and variance. While multiple works have analyzed BN for dense models, the role of layer normalization methods for Sparse Neural Networks (SNNs) has not been studied. In this work, we aim to understand the

impact of normalization layers, specifically BN, on training dynamics and convergence rate of SNNs; we theoretically and empirically show that layer normalization methods create training instability by changing the norm of gradients for SNN. Furthermore, we show that the operator norm of the Hessian during sparse training is significantly higher than that observed during dense training, indicating greater instability in the sparse training dynamics. To mitigate this, we propose a preconditioned gradient descent algorithm for sparse neural networks, which scales the gradient based on the sparsity/incoming connection of each neuron. Our contributions are as follows:

1. Our work is the first to theoretically analyze the impact of normalization layers on the gradient flow and training dynamics of SNNs. We derive the exact scaling factor required to account for the sparsity during backpropagation. To the best of our knowledge, our work is the first to systematically analyze the normalization layers for SNN.

2. We propose a new preconditioned gradient descent optimization method for sparse training and empirically show on multiple datasets and model architectures that our proposed optimization method improves the convergence rate of sparse training and generalization.

2. Background

Normalization Layers. DNNs are composed of stacked layers that apply linear mappings with learnable parameters followed by non-linear activation functions. While this architecture allows DNNs to learn complex and expressive features, it also introduces challenges such as vanishing or exploding gradients, which make deep networks difficult to train. To stabilize the training dynamics of deep networks, BN [11] was introduced and has become an essential component of DNNs. BN accelerates and stabilizes training by normalizing the layer's inputs through re-centering and re-scaling, effectively mitigating internal covariate shifts during training. Recent work has suggested that BN makes the optimization landscape significantly smoother, resulting in a more predictive and stable behaviour of the gradients, which facilitates faster training. Similar to BN, Layer Normalization (LN) was proposed by Ba et al. [1], which normalizes across the feature dimension instead of the batch dimension. Since LN does not rely on batch dimension, it has become a standard component of transformed-based models, including Large Language Models (LLMs).

Preconditioned Gradient Descent. Preconditioned gradient methods maintain a preconditioner matrix P, often the Hessian, for many popular optimizers, which is used to transform the gradient before taking an optimization step. A precondition gradient descent is defined by:

$$w^{t+1} = w^t - \eta P \nabla \mathcal{L}(w^t). \tag{1}$$

SGD is a special case of Equation (1), where the preconditioner P is an identity matrix. AdaGrad [4] uses the covariance matrix of the accumulated gradients as a preconditioner, Newton methods use the Hessian as a preconditioner. RMSProp uses an exponential moving average of gradients as a preconditioner. Recently proposed Shampoo optimizer [8] also falls under this category of optimization methods. The goal of preconditioning is to accelerate convergence, especially in optimization problems with ill-conditioned curvature.

Dynamic Sparse Training. Identifying effective sparse masks at initialization presents a significant challenge. Consequently, Pruning at Initialization (PaI) methods often fail to achieve dense model

performance, even at moderate sparsity levels. Recent theoretical studies underscore this limitation, indicating that PaI methods cannot match the performance of dense models [14]. In contrast to PaI, Dynamic Sparse Training (DST) methods address this limitation by iteratively updating the sparse mask during training. This dynamic adjustment allows DST methods to explore diverse sparse topologies, enabling performance comparable to that of dense models. A brief overview of DST methods is provided in the appendix.

Training stability. A number of different methods have been proposed to both understand and improve training dynamics. The initialization of parameters and the distributions of activations and gradients have been shown to be crucial for effective training, especially for very deep models. Glorot and Bengio [7] proposed the widely-used Xavier initialization, which initializes neural network weights so that the activations of each layer are approximately normally distributed. It has been shown that maintain the activation and gradient norm and variance stabilizes training dynamics by mitigating the issues of exploding and vanishing gradients. Building on this, Evci et al. [6] introduced a sparse variant of the initialization proposed by He et al. [10] that accounts for variations in the number of incoming connections (fan-in) for each neuron.

Edge of Stability Recent work has shown that gradient descent on dense neural networks typically operates in a regime termed "edge of stability"(EOS) where the operator norm (largest eigenvalue) of the Hessian oscillates around the critical threshold of $2/\eta$ and oscillates along the high-curvature directions without diverging [2]. In our work, we extend this phenomenon to sparse training. In particular, we find that when evaluated naively to DST, the operator norm of the Hessian is typically larger than the one we see in dense training. We conjecture that this contributes to training instability we witness in sparse training. We then show that our proposed gradient preconditioning method reduces the operator norm to roughly match that of the dense training.

3. Methodology

Motivation. Current state-of-the-art sparse training methods cannot yet match the generalization performance of dense models and often exhibit slower convergence rates [5, 17].It is possible that current normalization layers and optimizers are suboptimal for sparse training, as both significantly influence training dynamics and convergence rates. For instance, while normalization layers such as BN and LN have been extensively studied in the context of dense models, their impact on the training dynamics of SNNs remains largely unexplored. Motivated by this gap, in this work, we aim to deepen our understanding of sparse training dynamics and investigate how normalization layers (BN and LN) influence sparse training performance. Based on these insights, we propose a novel sparsity-aware optimizer to advance the state of sparse training.

3.1. Gradient Flow Analysis

Notations. Let $\mathcal{L}$ denote the loss function, and w represent the weight parameters. The sparsity mask is denoted by $\mathbf{M}$, which determines the active connections in the model. Let ℓ denote the ℓ-th layer of the model, x represent the preactivation inputs to the BN, and y represent the output of the BN. Finally, i denotes either the i-th neuron in a given layer or the i-th column vector in the weight matrix $\mathbf{W}$, depending on the context.

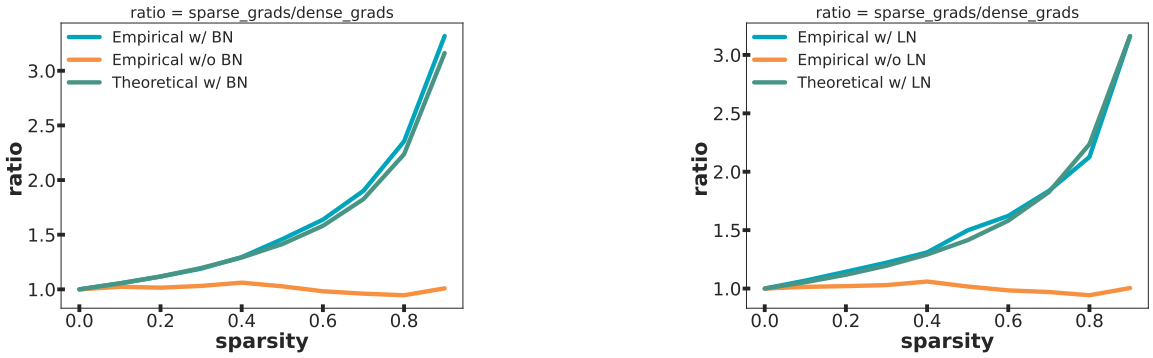


Figure 1: Effect of BatchNorm (left) and LayerNorm (right) on gradients. As observed, the gradients of a model with BN/LN increase with the increasing, creating instability in training.

Understanding the role of normalization layers. To understand the effect of masking, we analyze the gradient for the i-th feature/neuron in a layer ℓ just preceding BN , i.e., $\mathbb{E}\left[\frac{\partial f}{\partial x_i^{(\ell)}}\right]$, given by Equation (2):

$$\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_i^{(\ell)}}\right] = \frac{\gamma}{\sigma_i}\left[\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \hat{y}_i^{(\ell)}}\right] - \frac{1}{b}\sum_{j=1}^{m}\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \hat{y}_j^{(\ell)}}\right]\right], \tag{2}$$

where b denotes the batch. We derive the gradient flow for the SNN in Appendix B and observed that the gradients after the mask has been applied are scaled by a factor of $\sqrt{1/(1-s_i)}$ for the remaining weights, where s_i is the sparsity for the i-th neuron.

We perform a similar analysis for LN to understand its impact on the sparse training dynamics in Appendix B, and observed that LN increases the gradient by the same factor of $\sqrt{1/(1-s)}$. We empirically verify this using a two-layer network, as demonstrated in Section 3.1. We suspect this abrupt change in gradient direction introduces training instability, which may explain the slower convergence rate and worse generalization that dense training.

3.2. Proposed Method

For a stable training dynamics, masking should not scale the gradients for the remaining connections to avoid gradient exploding and better training dynamics. To this end, we propose a sparsity-aware preconditioned gradient descent optimization that takes into account the sparsity distribution of the layer.

Sparsity-aware Preconditioned Gradient Descent. As shown in Equation (5), gradients for the i-th neuron are scaled by a factor of $\sqrt{1/(1-s_i)}$, where s_i is the sparsity for the i-th neuron. One simple way to account for sparsity is to rescale the gradient flowing through that neuron by the inverse of the same factor, i.e., multiple the gradients of parameters in the i-th neuron by $\sqrt{(1-s_i)}$. We define the preconditioner matrix in Equation (1) as the diagonal matrix, where each diagonal element corresponds to the scaling factor for that parameter. Specifically, for the weight matrix $\mathbf{W}$, the scaling factor for parameters $\mathbf{W}[:,i]$ is given by $\sqrt{(1-s_i)}$. Mathematically, we define sparsity-aware gradient descent as:

$$w^{t+1} = w^t - \eta D \nabla \mathcal{L}(w^t), \tag{3}$$

where D is a diagonal matrix with diagonal elements equal to the corresponding scaling factor for the parameter. In Section 4, we validate our hypothesis on DST, which constantly updates the sparse mask during training and show that our preconditioner improves the generalization and convergence rate for DST.

4. Experimental Results

Results. To validate our hypothesis, we used CIFAR-100 [13] and ImageNet datasets [3]. We used ResNet-style architecture without the skip connection for our current experiments. We removed the skip connections to make the training dynamics easy to analyze. We used JaxPruner [15] to implement the RigL baseline and use the same hyperparameters as recommended by Evci et al. [5]. As shown in Table 1, at both moderate and higher sparsity, our method shows better generalization compared to the baseline on both CIFAR-100 and ImageNet datasets. We also plot the training loss and accuracy to show that our method converges at a faster rate in Figure 2 and Figure 3 (Appendix D). This empirically validates our hypothesis and show that the proposed preconditioned gradient descent improves the generalization of sparse models.

Table 1: Results on CIFAR-100 and ImageNet datasets. Our proposed sparsity-aware optimization method achieves better generalization than the baseline method at both moderate and high sparsity regimes.

Sparsity	**Method**	**Test Accuracy (%)**	
		CIFAR-100	ImageNet
0.80	Baseline	68.74 ± 0.1	65.25 ± 0.1
	Ours	$\mathbf{69.21 \pm 0.1}$	$\mathbf{66.13 \pm 0.4}$
0.90	Baseline	67.41 ± 0.4	64.19 ± 0.4
	Ours	$\mathbf{68.07 \pm 0.2}$	$\mathbf{64.55 \pm 0.5}$

Sparse training dynamics at the Edge of Stability We begin by looking at the training dynamics of dense training and sparse training. In Figure 4 (Appendix D), we plot the top 4 eigenvalues of the Hessian of the training loss. First, we see that the EOS phenomenon appears to hold for RigL with SGD but the largest eigenvalue of the Hessian is higher than that of the dense training. Second, we see that our proposed preconditioned SGD decreases the curvature of the Hessian to roughly match that of the dense training. We hypothesize that our preconditioner improves the training dynamics of RigL, though further analysis is required.

5. Conclusion and Future Work

Sparse training methods do not achieve the same generalization performance and convergence rates as their dense counterparts. Our work is the first one to study the effect of normalization layers on the sparse training dynamics. We theoretically and empirically demonstrated that common normalization layers, such as BN and LN, inadvertently destabilize sparse training by scaling up gradients for active connections. This phenomenon, we found, contributes to an increased operator norm of the Hessian, pushing sparse training further beyond the "edge of stability". To counteract this effect specifically for networks employing BN, we introduced a novel sparsity-aware optimization method. Our experiments confirm that this approach improve both generalization and the convergence rate in sparse training settings. In future work, we plan to extend our analysis and proposed methodology for models with residual connections and further explore its application with LayerNorm.

References

[1] Jimmy Lei Ba, Jamie Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. URL https://ieeexplore.ieee.org/abstract/document/5206848/.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[5] Utku Evci, Erich Elsen, Pablo Castro, and Trevor Gale. Rigging the lottery: Making all tickets winners, 2020. URL https://openreview.net/forum?id=ryg7vA4tPB.

[6] Utku Evci, Yani Ioannou, Cem Keskin, and Yann Dauphin. Gradient flow in sparse neural networks and how lottery tickets win. In *Proceedings of the AAAI conference on artificial intelligence*, 2022.

[7] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

[8] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

[14] Tanishq Kumar, Kevin Luo, and Mark Sellke. No free prune: Information-theoretic barriers to pruning at initialization. In *International Conference on Machine Learning*, pages 25662–25681. PMLR, 2024.

[15] Joo Hyung Lee, Wonpyo Park, Nicole Elyse Mitchell, Jonathan Pilault, Johan Samir Obando Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, Woohyun Han, Shivani Agrawal, Suvinay Subramanian, Xin Wang, Sheng-Chun Kao, Xingyao Zhang, Trevor Gale, Aart J.C. Bik, Milen Ferev, Zhonglin Han, Hong-Seok Kim, Yann Dauphin, Gintare Karolina Dziugaite, Pablo Samuel Castro, and Utku Evci. Jaxpruner: A concise library for sparsity research. In *Conference on Parsimony and Learning (Proceedings Track)*, 2023. URL https://openreview.net/forum?id=H2rCZCfXkS.

[16] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

[17] Bowen Lei, Dongkuan Xu, Ruqi Zhang, Shuren He, and Bani Mallick. Balance is essence: Accelerating sparse training via adaptive gradient correction. In *Conference on Parsimony and Learning*, pages 341–378. PMLR, 2024.

[18] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[19] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Appendix A. Related Work

Sparse Training from Initialization. PaI methods remove or prune weights using heuristics to create a sparse mask, which is subsequently used for training a sparse model. Various heuristics for pruning weights have been proposed (discussed below). However, identifying an effective sparse mask at initialization remains a significant challenge, and no PaI method has yet matched the performance of dense models. **Random Pruning** assigns importance scores to network weights by sampling from a uniform distribution. Weights with the lowest scores are then pruned to achieve the desired level of sparsity. This method acts as a naive baseline, as any effective pruning approach is expected to outperform random selection. **Magnitude Pruning** removes weights based on their L1 norm ($|w_i|$), and despite its simplicity as a heuristic, it is the most commonly used heuristic [9]. Lee et al. [16] proposed **SNIP**, a method designed to identify weights with the greatest impact on the loss function. SNIP computes the gradients g of the loss function with respect to each weight parameter w using a subset of the training data. The pruning score is calculated as the product of the gradient and the weight's magnitude, i.e., $|g_i \cdot w_i|$. Wang et al. [19] proposed **GraSP**, which leverages second-order gradient information and the Hessian to determine weight importance. GraSP computes the Hessian-gradient product h_l for each layer l and assigns the pruning score as $s_l = -w_l \odot h_l$. The method aims to prune weights that do not affect the gradient flow while preserving those crucial for maintaining effective gradient propagation.

Sparse Evolutionary Training (SET). To address the limitations of training with a static sparse mask, Mocanu et al. [18] proposed SET, which initiates training with a randomly generated sparse mask. Since a random sparse mask cannot capture the relationship between the data and the model, a fraction of weights with the lowest magnitudes are removed after a few epochs or steps to maintain the sparsity level. Simultaneously, an equal number of new connections are randomly activated. This dynamic pruning and regrowth of connections allows SET to maintain stochasticity during optimization, achieving better performance than PaI methods. SET is inspired by biological neural networks, particularly the synaptic pruning that occurs during sleep, where weaker synapses are eliminated.

Rigging the Lottery Ticket (RigL). Inspired by SET, RigL, proposed by Evci et al. [5], does not start with a random mask that has uniform layer-wise sparsity. Instead, it uses the Erdős-Rényi-Kernel (ERK) distribution, which considers the channel dimensions and layer widths to generate the initial random sparse mask. Similar to SET, RigL updates the mask Similar to SET, RigL updates the mask during training. However, unlike SET, which grows connections randomly, RigL grows connections based on the gradient norm. Specifically, pruned or masked parameters with the highest gradient values are reactivated. Although RigL achieves performance comparable to dense models, its slower convergence leads to a training FLOP count comparable to that of dense training. The criterion for updating masks in RigL is as follows:

1. Update Schedule: RigL uses a cosine decay schedule to update the mask. Let ΔT denote the number of iterations between each update step, T_{end} denote the iteration at which to stop updating the mask, and α the initial fraction of mask parameters to be updated, then the cosine schedule can be given by:

$$f_{decay}(t;\alpha,T_{end}) = \frac{\alpha}{2}\left(1+\cos\left(\frac{t\pi}{T_{end}}\right)\right) \tag{4}$$

2. Drop Criterion: At each mask update step, RigL removes an f fraction of the weights based on their magnitudes.

3. Grow Criterion: Unlike SET, which reactivates connections randomly, RigL uses dense gradient information to activate connections with the highest gradient norm. Consequently, RigL requires access to dense gradients during each mask update.

Appendix B. Understanding Batch Normalization

Effect of sparity on feature variance. Let h denote the output of the previous layer, fan_{in} be the original number of incoming connections to a neuron, and w_i denote the weights of the ith column in the weight matrix $\mathbf{W}$. The change in feature statistics when a mask $\mathbf{M}$ (with sparsity s_i) is applied can be calculated as:

$$
\begin{aligned}
x_i &= h^T w_i = \sum_j h_j^T w_{ij} \\
x_i &= h^T (w_i \odot M_i) + h^T (w_i \odot (1 - M_i)) \\
\sigma^2 &= \text{var}(x_i) = \text{fan}_{\text{in}} * \text{var}(h_j * w_i) \\
x_i^{'} &= h^T (w_i \odot M_i) \\
\therefore \sigma^{'2} &= \text{var}(x_i^{'}) = (1 - s_i) * \text{fan}_{\text{in}} * \text{var}(h_j * w_{ij})
\end{aligned}
$$

Thus, masking decreases the variance of features by a factor $1/\sqrt{1-s_i}$.

Effect of masking on the gradient flow. The impact of this change in feature variance on the gradient flow can be mathematically analyzed as follows:

$$
\begin{aligned}
\left(\mathbb{E}\left[\frac{\partial f}{\partial x_i^{'}} \right] \right)_{\text{sparse}} &= \frac{\gamma}{\sigma^{'} * b} \left[\mathbb{E}\left[\frac{\partial f}{\partial \hat{y}_i} \right] - \frac{1}{b} \sum_{j=1}^{m} \mathbb{E}\left[\frac{\partial f}{\partial \hat{y}_j} \right] \right] \\
&= \frac{1}{\sqrt{(1 - s_i)}} \left(\mathbb{E}\left[\frac{\partial f}{\partial x_i} \right] \right). \qquad (5)
\end{aligned}
$$

As observed in Equation (5), the gradients after the mask has been applied are scaled by a factor of $\sqrt{1/(1-s_i)}$ for the remaining weights. This abrupt scaling introduces training instability and changes the gradient direction, potentially explaining the slower convergence rate observed in sparse training.

Appendix C. Understanding Layer Normalization

Effect of masking on LayerNorm. Following a similar analysis for LN, we derive the gradients for each neuron as follows:

$$\frac{\partial L}{\partial \gamma} = \sum_i \frac{\partial L}{\partial y_i} \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$
$$\frac{\partial L}{\partial \beta} = \sum_i \frac{\partial L}{\partial y_i}$$
$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j} \left[\gamma \left(\frac{N\delta_{ij} - 1}{N\sigma} - \frac{(x_i - \mu)(x_j - \mu)}{\sigma^3} \right) \right],$$

where L is the loss function, y_i is the ith normalized feature, x_i is the unnormalized feature, δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$, 0 otherwise), and N is the number of elements in the input vector. For each iteration, assuming the features are i.i.d. distributed, i.e., $\mathbf{cov}(x_i, x_j) = 0 \forall\, i \neq j$, gradients averaged over a batch can be written as:

$$\begin{aligned}
\mathbb{E}[\frac{\partial L}{\partial x_i}] &= \mathbb{E}\frac{\partial L}{\partial y_i} \left[\gamma \left(\frac{N-1}{N\sigma} - \frac{(x_i - \mu)(x_i - \mu)}{\sigma^3} \right) \right] \\
&\quad + \mathbb{E} \sum_{j \neq i} \frac{\partial L}{\partial y_j} \left[\gamma \left(\frac{-1}{N\sigma} - \frac{(x_i - \mu)(x_j - \mu)}{\sigma^3} \right) \right] \\
&= \frac{\partial L}{\partial y_i} \left[\gamma \left(\frac{N-1}{N\sigma} - \frac{\mathrm{var}(x_i)}{\sigma^3} \right) \right] \\
&\quad + \sum_{j \neq i} \frac{\partial L}{\partial y_j} \left[\gamma \left(\frac{-1}{N\sigma} - \cancel{\frac{\mathbf{cov}(x_i, x_j)}{\sigma^3}} \right) \right] \\
&= \frac{\partial L}{\partial y_i} \left[\gamma \left(\frac{N-1}{N\sigma} - \frac{\mathrm{var}(x_i)}{\sigma^3} \right) \right] \\
&\quad + \sum_{j \neq i} \frac{\partial L}{\partial y_j} \left[\gamma \left(\frac{-1}{N\sigma} \right) \right]
\end{aligned} \tag{6}$$

When a sparse mask with sparsity s and density $(1 - s)$ is applied, σ^2 and $\mathrm{var}(x_i)$ changes as follows:‘

$$\mathrm{var}(x_i)_m = (1 - s) \times \mathrm{var}(x_i) \tag{7}$$
$$\sigma^2 = \sum_i \mathrm{var}(x_i)$$
$$\therefore \sigma_s^2 = (1 - s) \times \sum_i \mathrm{var}(x_i) \tag{8}$$

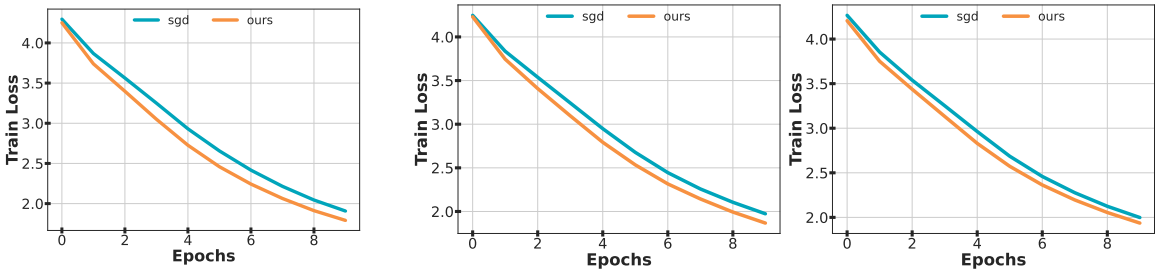
Plugging in new σ and $\mathrm{var}(x_i)$ from Equations (7) and (8) into Equation (6), sparse gradients with LN can be derived as:

$$\begin{aligned}
\mathbb{E}_s[\frac{\partial L}{\partial x_i}] &= \frac{\partial L}{\partial y_i}\left[\gamma\left(\frac{N-1}{N\sigma_m} - \frac{\mathrm{var}_m(x_i)}{\sigma_m^3}\right)\right] \\
&= \frac{\partial L}{\partial y_i}\frac{\gamma}{\sigma_m}\left[\frac{N-1}{N} - \frac{\mathrm{var}_m(x_i)}{\sigma_m^2}\right] \\
&= \frac{\partial L}{\partial y_i}\frac{\gamma}{\sqrt{(1-s)\sigma^2}}\left[\frac{N-1}{N} - \frac{\mathrm{var}(x_i)}{\sigma^2}\right] \\
&+ \frac{1}{\sqrt{(1-s)}}\sum_{j\neq i}\frac{\partial L}{\partial y_j}\left[\gamma\left(\frac{-1}{N\sigma}\right)\right]
\end{aligned} \tag{9}$$

Therefore, the sparse gradient $\mathbb{E}_s[\frac{\partial L}{\partial x_i}]$ is scaled by a factor of $\sqrt{1/(1-s)}$.

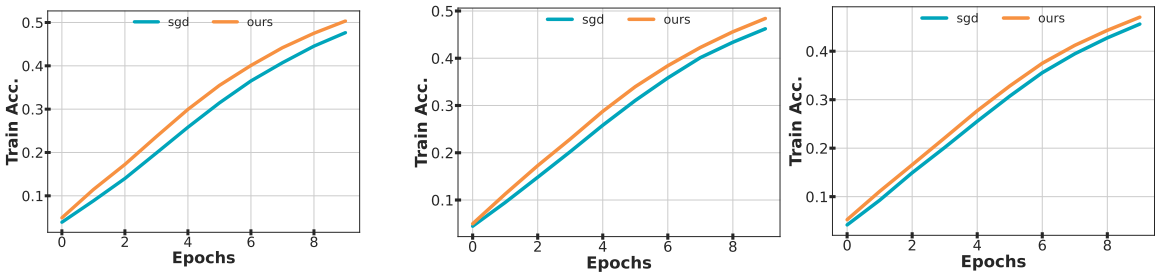
Appendix D. Additional Plots

We plot the evolution of training loss and training accuracy in Figure 2 and Figure 3 to demonstrate that our proposed method improves sparse training dynamics. We provide additional results on the curvature/edge of stability in Figure 4.



(a) **sparisty=0.8** (b) **sparisty=0.9** (c) **sparisty=0.95**

Figure 2: **Evolution of training loss**. As shown, our proposed method has better sparse training dynamics and convergence rate, specifically early in the training the loss decreases at a faster rate. This empirically shows that training with the proposed preconditioned gradient descent improves sparse training dynamics.



(a) **sparisty=0.8** (b) **sparisty=0.9** (c) **sparisty=0.95**

Figure 3: **Evolution of training accuracy**. As shown, our proposed method has better sparse training dynamics and convergence rate, training accuracy increases at a faster rate with our proposed method.

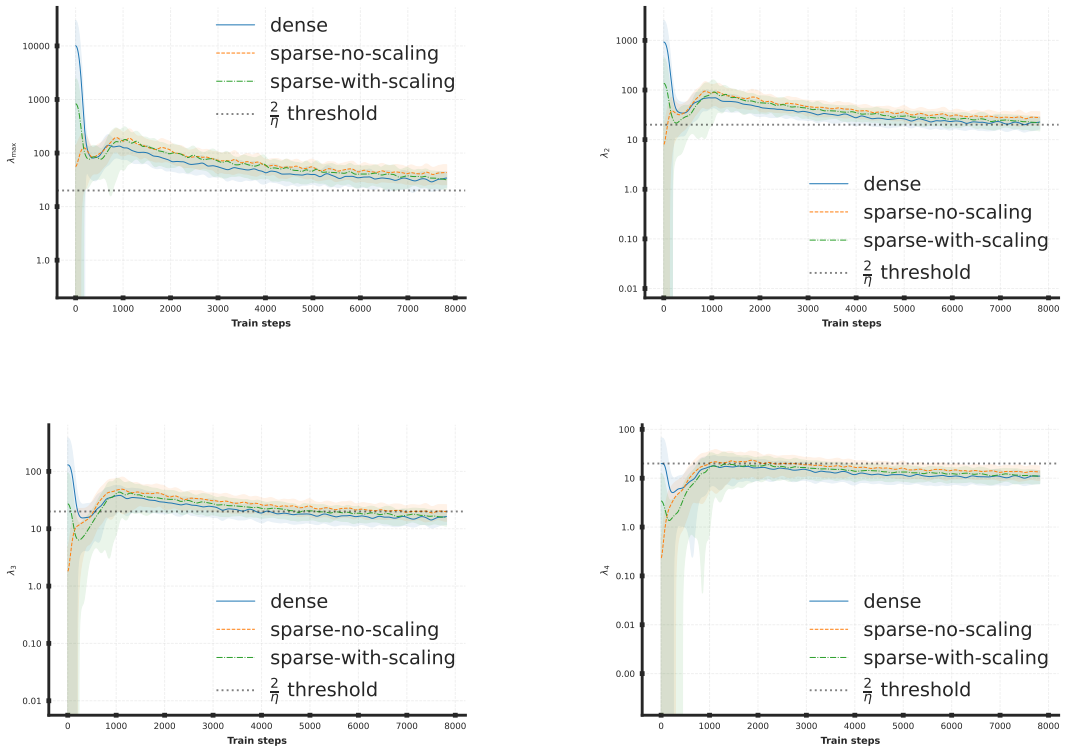


Figure 4: **Edge of Stability.** This plot illustrates the evolution of the maximum eigenvalue ($\lambda_{\max}$) of the Hessian of the training loss across different training regimes, averaged over 4 runs. The solid blue line represents dense training, the orange dashed line depicts sparse training without scaling, and the green dot-dashed line shows sparse training with scaling. The shaded area denotes the 90% confidence interval. The dotted gray line indicates the $2/\eta$ EOS threshold.