The Price of Sparsity: Generalization and Memorization in Sparse Neural Networks

Ziyu Ye, **Chaoqi Wang**, **Zixin Ding**, and **Yuxin Chen**
University of Chicago
{ziyuye, chaoqi, zixin, chenyuxin}@uchicago.edu

Abstract

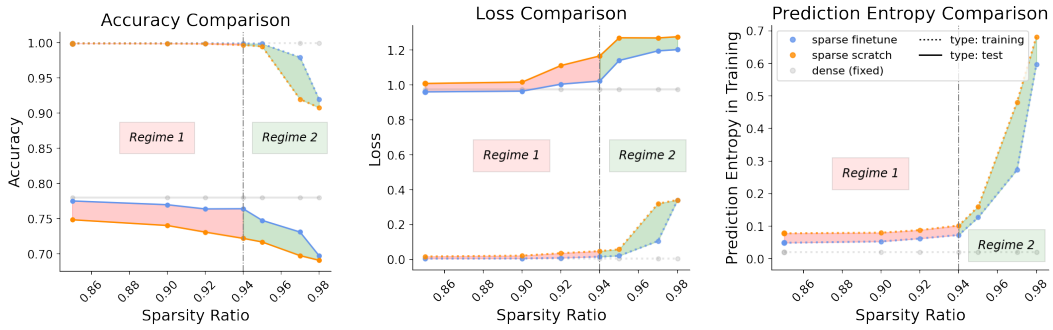
The future of deep learning is sparse: by inducing sparsity in neural networks, we are able to train models at scale with unprecedented efficiency, paving the way for the next-generation AI architectures. However, recent works show that the solutions found by sparse training from scratch (`Sol-S`) may face inevitable performance degradation compared to those by sparse finetuning (`Sol-F`), casting a pall over the true efficiency of sparse neural networks. In this paper, we put forward an extensive empirical study on this important and foundational issue. We first observe that `Sol-S` can be categorized into two distinct regimes, which we termed as the *generalization regime* and the *optimization regime*. With analysis on Fisher information, we provide a unified explanation on the underlying mechanism of the two regimes: *sparse neural networks trained from scratch are weaker at memorization, and require more information in learning*; this mechanism entangles `Sol-S` with sharper local minima and higher sensitivity in the generalization regime and memorization failures in the optimization regime (yet it may bring better noise robustness). Based on our findings, we propose insights on strategies to improve the performance of sparse training from scratch via loss regularization and data scheduling. We hope our discoveries can fuel future work to understand and improve the trainability and generalizability of deep and sparse networks.

1 Introduction: A Tale of Two Regimes

Scaling up deep learning models has been demonstrated to be an effective and reliable way to improve the performance across a broad range of tasks [14, 6, 13, 54]. However, training such over-parameterized models is costly. For example, training a GPT-3 model will cost twelve millions dollars and produce over five-hundred tons of CO_2 equivalent emissions [44]. Recently, training a sparse network (*i.e.*, a network with most parameters being zero) has emerged as a promising direction to reduce the training cost and improve the inference efficiency [25]. The major ways to find a sparse solution (*i.e.*, a converged sparse network parameter) include sparse training by finetuning (*sparse finetuning*, hereafter) and sparse training from scratch (*sparse scratch*, hereafter). As a classical approach, sparse finetuning works by re-using the trained parameters of a dense network and finetuning upon them; since it requires dense training, this approach is not genuinely efficient. In contrast, sparse scratch works by generating a sparse mask prior to training, and initialize parameters from scratch [33, 53, 51], enjoying superior efficiency over the former. However, the big challenge is, that there exists a **performance gap** (or generalization discrepancy, which we will use interchangeably) between sparse scratch and sparse finetuning. In other words, the solution of sparse scratch (*i.e.*, `Sol-S`) has significantly worse *test accuracy* than that of the solution of sparse finetuning (*i.e.*, `Sol-F`) [16, 21]. Considering the remarkable benefits of sparse scratch, we ask:

What is the root cause for the performance gap? How may we close it?

Work in progress. Do not distribute.



(a) (b) (c)

Figure 1: **The *generalization regime* and *optimization regime* of sparse scratch**. The plots show the performance of the final solutions of sparse training from scratch (Sol-S, orange line) and sparse training by finetuning (Sol-F, blue line) on CIFAR-100 with ResNet32 on varying sparsity ratios. Experimentation details are specified in Appendix. (a) **Accuracy**. In regime 1, there exist large generalization discrepancies (*i.e*, the pink shaded area between thick lines) for Sol-F and Sol-S while they maintain almost the same near-optimal training accuracy. We refer to this regime as the *generalization regime*. In regime 2, there emerge large discrepancies on both training and test accuracy (*i.e.*, the green shaded area). We refer to this regime as the *optimization regime*. (b) **Loss**. Similar patterns exist in regime 1 and regime 2. (c) **Training prediction entropy**. Clearly, Sol-S has a higher prediction uncertainty and there exists a sharp phase transition from regime 1 to regime 2.

Though there are some initial attempts [16, 50, 21] to understand the performance gap between sparse scratch and sparse finetuning, it is still unclear what the fundamental mechanism is underneath, and how to improve sparse training based on the findings. To answer these questions, we provide a more fine-grained study on the performance gap and seek to shed lights on remedying the performance gap. We find that there exist two regimes of sparse scratch as shown in Figure 1, where the generalization discrepancy exists but under fairly different mechanisms. Specifically, in regime 1, the training performance of Sol-S can match that of Sol-F and is near-optimal; however, as the sparsity ratio increases,[1] Sol-S suffers an optimization failure, *i.e.*, it has considerably worse training accuracy compared to Sol-F. We name these two regimes as the generalization regime and the optimization regime, respectively. We summarize our key contributions below.

- First in literature, we identify and characterize the performance gap from sparse scratch to sparse finetuning by two regimes: the generalization regime and the optimization regime.
- We provide a unified explanation for the two regimes: *sparse scratch are weaker at memorization, and requires more information in learning*. By this, we identify the driven factors for the worse performance of sparse scratch in the generalization regime (*i.e.*, sharper local minima and higher sensitivity) and the optimization regime (*i.e.*, weaker memorization).
- Based on our findings, we provide insights on several simple strategies to improve the performance of sparse neural networks via loss regularization or data scheduling.

2 Preliminaries

Notations. Let $\mathbf{x} \in \mathbb{R}^m$ denote an input (*e.g.*, an image) sampled from a probability distribution $p(\mathbf{x})$, $y \in \mathbb{Y} = \{1, \cdots, C\}$ be the class label sampled from $p(y|\mathbf{x})$ for each input $\mathbf{x}$, and $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote the training dataset of size N. A neural network f parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ encodes a conditional distribution $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. The objective is to minimize the empirical risk $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \ell(\mathbf{x}_i, y_i; \boldsymbol{\theta})$, where $\ell(\cdot, \cdot)$ denotes the cross-entropy loss in our case. Given a network $f(\cdot; \boldsymbol{\theta})$ with a binary mask $\boldsymbol{m}$, we denote $f(\cdot; \boldsymbol{\theta} \odot \boldsymbol{m})$ as the resulted sparse network.

[1] While the two regimes here may be naïvely separated by the sparsity ratio, there may exist more nuances to be discovered by future works.

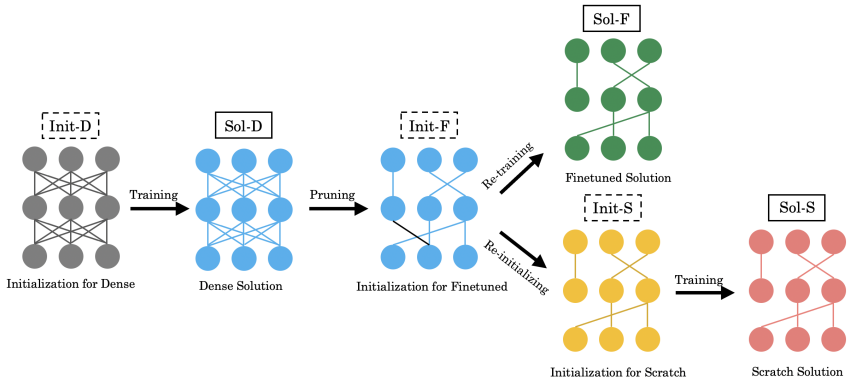


Figure 2: **Training procedures.** We start with a densely connected network whose parameters are denoted as `Init-D` and the trained **dense solution** is denoted as `Sol-D`. To find a sparse mask, we prune `Sol-D` and the remaining parameters are denoted as `Init-F`. We consider two types of sparse solutions: (1) the **finetuned solution** `Sol-F`, which is obtained by training with `Init-F`; (2) the **scratch solution** `Sol-S`, which is obtained by training `Init-S` that comes from randomly initializing parameters of `Init-F`; for simplicity, we re-use the sparse mask `Init-F`, as similarly done in [16].

Sparse training procedures. We consider two approaches for training sparse neural networks: sparse training from scratch (i.e., *sparse scratch*), and sparse training by finetuning (i.e., *sparse finetuning*). The procedures and notations are specified in Figure 2. We use one-shot magnitude pruning on a trained dense network to find the sparsity mask [58].[2] This strategy works by removing the weights with the least ℓ_1-norm or absolute value, which is equivalent to the optimal brain damage [32]. Though being simple, this strategy has shown to be a solid pruning strategy [36, 16], and introduces minimal extra complexity to the empirical setup.

Hessian matrix and the loss curvature. The Hessian matrix is defined as the second derivative of the loss function $\mathcal{L}(\cdot)$ with respect to the network parameter $\boldsymbol{\theta}$, we denote it as

$$\mathbf{H}(\boldsymbol{\theta}) = \nabla^2_{\boldsymbol{\theta}}\, \mathcal{L}(\boldsymbol{\theta}). \tag{2.1}$$

From a loss landscape perspective, numerous works have analyzed the relationship between the loss Hessian and the generalizability or trainability of neural networks [15, 52, 22]. The intuition is that a more degenerate Hessian matrix entails a flatter region for the local minima, while a flatter minima is more tolerant on the data distributional shift, leading to better generalization [29, 42].

Jacobian matrix and sensitivity. Given an input $\mathbf{x}$, the Jacobian matrix on the parameter $\boldsymbol{\theta}$ is:

$$\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}^\mathrm{T}} f(\mathbf{x}; \boldsymbol{\theta}). \tag{2.2}$$

The input-output Jacobian matrix measures the local sensitivity of neural networks around the input. In general, a higher sensitivity (or higher spectral norm of the Jacobian matrix) implies worse generalization [43][3] as it is less tolerant to the input noise or data distribution shift.

Memorization in deep neural networks. Recent works have empirically demonstrated the impact of *label memorization* on the generalizability or trainability of neural networks [4, 9, 56, 49]. Specifically, the Long Tail Hypothesis states that *memorization of data labels is necessary for achieving near optimal generalization error* on a long-tailed data distribution (e.g., CIFAR-10, ImageNet, etc.) [18, 19, 5]. This phenomenon may also be referred to as "benign overfitting" [7], and different architectures have shown to possess different inductive bias towards memorization [55].

[2] Many related works also discuss the lottery setting [20, 16] (different from the sparse training from scratch, it re-uses the parameters from `Init-D` as the initialization) and multi-shot or dynamic sparse training strategies [40, 37, 17]. We plan to incorporate comparisons with those settings in the future revisions.

[3] In this paper, we use the mean squared Frobenius norm of the Jacobian matrix evaluated on the training set (*i.e.*, $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\|\mathbf{J}(\mathbf{x})\|_F^2]$, or Jacobian norm, hereafter) as the sensitivity measure, as shown in Table 2.

Fisher information. We use Fisher information to measure the information that the training set carries about the network parameter $\boldsymbol{\theta}$, which is formally defined as the covariance of the gradient of the log likelihood estimate of the training set:

$$\mathbf{F}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\hat{y}\sim p_{\boldsymbol{\theta}}(y|\mathbf{x})}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\hat{y} \mid \mathbf{x}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\hat{y} \mid \mathbf{x})^T\right]. \tag{2.3}$$

By definition, Fisher information measures the confidence of the network on its solution given the training data. It has intrinsic connections to the loss curvature and sensitivity of the neural network. In this paper, we also claim that it is closely related to networks' memorization ability.[4] Specifically:

- Relationship to the loss curvature: As proved in [38], $\mathbf{F}(\boldsymbol{\theta})$ can be seen as a semi-definite approximation of $\mathbf{H}(\boldsymbol{\theta})$; for a well-trained network (i.e., almost all training data are correctly predicted), by the first-order approximation, we have $\mathbf{H}(\boldsymbol{\theta}) \approx \mathbf{F}(\boldsymbol{\theta})$.
- Relationship to the sensitivity: Assuming perturbing the network parameter $\boldsymbol{\theta}$ by δ such that $\boldsymbol{\theta}' = \boldsymbol{\theta} + \delta\boldsymbol{\theta}$, the change in the network's output can be represented by $\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\mathrm{KL}\left(p_{\boldsymbol{\theta}'}(y \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(y \mid \mathbf{x})\right) = \delta\boldsymbol{\theta}^\top \mathbf{F} \delta\boldsymbol{\theta} + o\left(\|\delta\boldsymbol{\theta}\|_2^2\right)$. Thus, $\mathbf{F}(\boldsymbol{\theta})$ can be seen as a measure for the *parameter sensitivity* of a neural network; this property may also be referred to as "effective connectivity" or "synaptic strength" which impacts generalization [2, 30, 3].
- Relationship to data memorization: Some initial attempts show that Fisher information is relevant to data memorization, which in turn impacts the networks' generalization ability [2, 3, 39, 28]. Here, we conjecture that a *higher Fisher information associated with some samples implies that such samples are harder for the network to memorize*, such that Fisher information can be considered as a proxy for the oddness or difficulty of training examples.

We will use the above interpretations on Fisher information[5] to analyze the trainability and generalizability of sparse neural networks in the following sections.

3 Experimental Results

We here present our results on understanding the two regimes of performance gap. Following the empirical validation on the two regimes in Figure 1 and the trend of $\mathbf{Tr}(\mathbf{F}(\boldsymbol{\theta}))$ in Figure 3, we provide a high-level summary on the underlying mechanism in Table 1, which will be elaborated next.

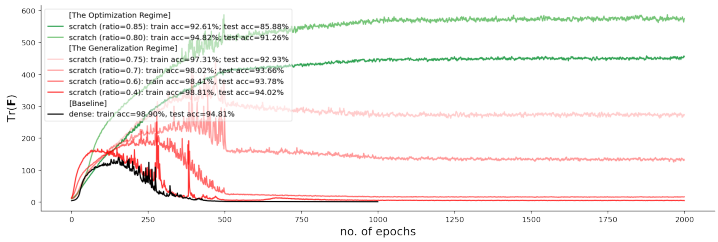


Figure 3: **Trend of Fisher information.** This plot provides a rough intuition on the relationship of the two regimes with the trend of Fisher information. We present the trajectory of the $\mathbf{Tr}(\mathbf{F}(\boldsymbol{\theta}))$ by training on the synthetic dataset in [47] using MLP with varying sparsity ratio. Loosely speaking, in the generalization regime, the Fisher information of the sparse scratch first increases, then gradually drops at a moderate level higher than the baseline's; this implies that the networks may have fitted the training data, but fallen in a sharper minima with higher sensitivity. In the optimization regime, the Fisher information increases and hardly drops; specifically, there may exist some *hard examples* in the training dataset that the networks cannot fit or memorize, such that it will consistently overfit and underfit partially during the stochastic gradient update, causing the Fisher information going up; this weak memorization may lead to the optimization failure in this regime.[6]

[4] We use Fisher trace $\mathbf{Tr}(\mathbf{F}(\boldsymbol{\theta})) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\hat{y}\sim p_{\boldsymbol{\theta}}(y|\mathbf{x})}\left[\|\nabla_{\boldsymbol{\theta}}\, \ell(\mathbf{x}_i, y_i; \boldsymbol{\theta})\|_2^2\right]$ as a proxy to measure $\mathbf{F}(\boldsymbol{\theta})$.

[5] There are also discussions on the connections of Fisher information to the (Shannon) Information Bottleneck Theory and the stochastic learning process [47, 1, 10]. We plan to provide additional experiments and analysis on this in the appendix and in future revisions.

[6] Be aware that this is a work in progress and it has some deficits (*e.g.*, the training performance in the generalization regime is not close to 100%, such that it slightly deviates from our definition of this regime where the training accuracy should be near optimal; also, the results on sparse finetuning are omitted here due to time constraints). In future revisions, we will update this figure with results on CIFAR-100 using ResNet32.

	Training Error	Trend of Fisher Information	Reasons for the Generalization Discrepancy
Generalization Regime	near optimal	increases first, then decreases, yet ends up with a higher level compared to sparse finetuning's	sharper minima, higher sensitivity
Optimization regime	far worse than the optimal	increases and hardly decreases	weaker memorization on training data

Table 1: Summary of the underlying mechanisms for sparse training from scratch.

3.1 The Generalization Regime: *The Curse of Information*

Here, we presented evidence on the underlying mechanism for the generalization regime. As the sparsity ratio increases, the network will require more information in learning, such that the *trace of Fisher information* increases. As shown in Section 2, this implies that the network may fall into *sharper local minima* (which may be measured by $\mathbf{Tr}(\mathbf{F})$) with *higher sensitivity* (which may be measured by Jacobian norm). Specifically, as shown in Table 2, `Sol-S` possesses much larger $\mathbf{Tr}(\mathbf{H})$, $\mathbf{Tr}(\mathbf{F})$ and Jacobian norm than `Sol-F`. We refer this phenomenon as **the curse of information**.

	Sparsity Ratio	$\mathbf{Tr}(\mathbf{F})$	$\mathbf{Tr}(\mathbf{H})$	Jacobian Norm
`Sol-D`	0.00	53.81	66.32	35.60
`Sol-F`	0.90	543.27	720.49	43.27
	0.94	2573.51	4198.26	52.25
	0.96	4376.32	9618.40	67.92
`Sol-S`	0.90	2701.68 ↑	10452.93 ↑	63.74 ↑
	0.94	9840.60 ↑	12905.88 ↑	66.36 ↑
	0.96	15818.65 ↑	20715.41 ↑	70.10 ↑

Table 2: **Fisher information, loss curvature, and sensitivity.**[7] The experimental setup of this table is the same as in Figure 1 on CIFAR100 with ResNet32. We use Monte-Carlo Sampling on the training set to estimate $\mathbf{Tr}(\mathbf{F})$ and Hutchinson's method to estimate $\mathbf{Tr}(\mathbf{H})$ as in [27].

3.2 The Optimization Regime: *Memorization as a Double Edged Sword*

In the optimization regime, for a fixed sparsity ratio, `Sol-S` has a much worse optimization error (or training accuracy) compared to that of `Sol-F`, leading to the generalization discrepancy in this regime. While many works have attributed the optimization failure by capacity issue [4, 23], it cannot explain for our case, as `Sol-S` and `Sol-F` have the same structure capacity (*i.e.*, the architecture are the same) and neuron capacity (*i.e.*, the weight connectivity of neurons are the same).

As discussed in Section 2, the Long Tail Hypothesis states that many real-world data distributions contain long-tailed or less frequent data that requires memorization in order to have good performance on the test examples from their sub-populations. Here, we propose the **Memorization Hypothesis for Sparse Training** as the fundamental reason for the performance gap in the optimization regime: *sparse scratch are weaker at memorization than sparse finetuning*.

To explain, since the initialization of sparse finetuning inherits weights from a well-trained densely connected network, it may already entail some memorization on data labels (especially for those hard examples). However, sparse scratch initializes its weights randomly, and since memorization happens only at the late phase of training [35], it is much harder for `Sol-S` to achieve the same level of memorization as that of `Sol-F`, causing the phenomenon of optimization failure in this regime.

To verify our hypothesis, instead of directly training on a natural data distribution, and verify if those examples that `Sol-F` and `Sol-S` has disagreement on are truly those long-tailed examples that

[7] As a work in progress, we here only test 3 different sparsity ratios. In addition to those presented indicators, another promising direction is analyze the local smoothness of the loss function for sparse scratch, which is shown to be a crucial factor for generalization [41]. Notice that those of sparsity ratio of 0.96 in effect correspond to the optimization regime; we incorporate it here to illustrate that the narrative of sharper minima and higher sensitivity is still valid for this regime, but they may not be the most fundamental driven factor.

requires memorization (which is computationally expensive [31, 19]),[8] we adopt a noisy training setting, and shows that `Sol-S` memorize less on those noisy data, as demonstrated in Figure 4.

Here, memorization can be seen as a double-edged sword since in this case, `Sol-S` reaches better test performance (around 12% better than that of `Sol-D` and 10% better than that of `Sol-F`). We call this phenomenon as *benign obliviscence* as it endows `Sol-S` with better robustness.[9] Our results may also shed lights on the barrier between `Sol-F` and `Sol-T` as discussed in [34, 16].

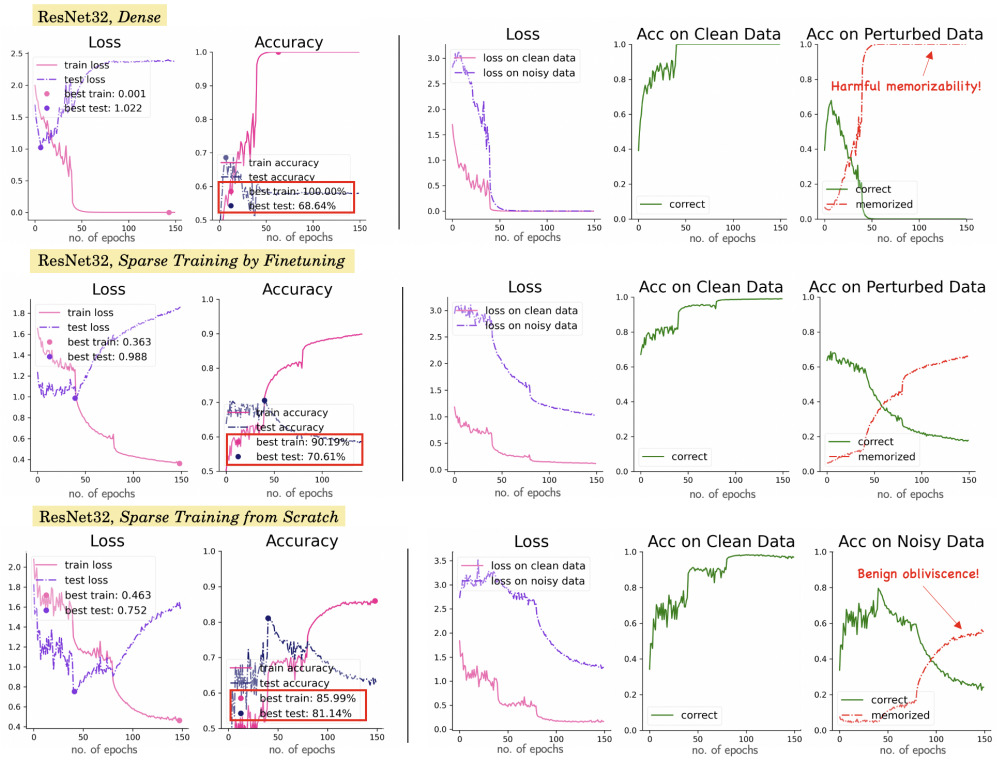


Figure 4: **Sparse training from scratch is more robust to label noise.** This experiment is conducted on CIFAR-10 with ResNet32. The training set contains 30% perturbed data whose labels are uniformly randomly flipped as similarly conducted in [35]. The sparsity ratio is 0.95 for sparse training. The learning rate is 0.02 and decay by 0.1 at the 40th and 80th epoch. The left two columns show the performance on the training set (noisy) and test set (clean), and the right two columns show the performance on the clean and noisy data in the training set; the notation `correct` means predicted label equals to true label, while `memorized` means predicted label equals to noisy label. Clearly, sparse scratch is weaker at memorizing the data since it has the lowest training accuracy, and it starts to memorize at a much later stage in training.

4 Insights on Potential Algorithms for Closing the Gap

4.1 The Loss Regularization Perspective

Knowing that `Sol-S` has worse generalization performance due to the curse of information (which leads to sharper minima and higher sensitivity) and the weak memorization ability, a direct intuition is to add regularization term in the loss function in order to push for a flatter-minima solution with less sensitivity, and encourages data memorization. There is some prior work on this direction, such as [8] which directly pursues a flat minimum valley in the training loss. Given our analysis, it might be convenient to directly control the Fisher information; the work mostly align to this might be [28] which proposes a loss regularizer with respect to Fisher information. However, they apply

[8] As a side note, [26] shows that the pruned network has disagreement on the dense network mostly on those less-frequent and long-tailed classes. See appendix for our discussion for `Sol-F` and `Sol-S`.

[9] Some recent work also discuss the robustness of sparsity in the adversarial setting [12, 24].

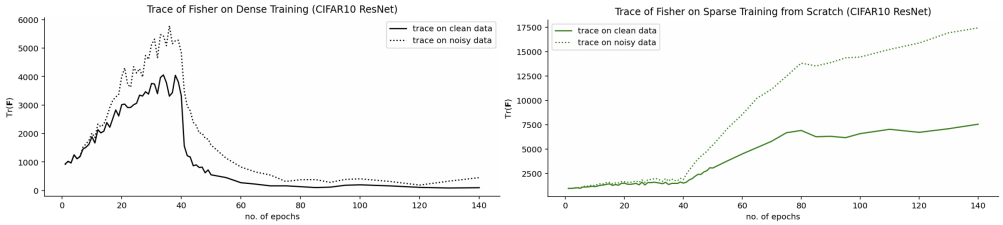


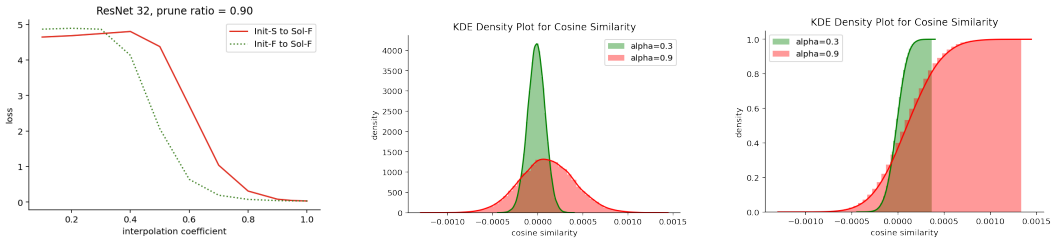
Figure 5: **Trend of Fisher information.** The experimental setup is the same as in Figure 4. As a supplementary figure, the results demonstrate that the Fisher information is higher for those noisy data, and sparse scratch has a higher Fisher information which hardly decrease when the network can not memorize all the training data.

the regularization on over-capacity dense networks and aim to reduce (harmful) memorization. In our case, we would like to moderately *encourage the memorization* for sparse neural networks. A potential easy fix is to adaptively apply negative Fisher regularization at the initial epochs (in order to encourage memorization), and remove it at the later epochs to avoid unnecessary exploitation.

4.2 The Data Schedule Perspective

In [16], the authors show that their exists a linear path between `Init-S` and `Sol-F`, where the loss on the training data is monotonically decreasing. We compute the training loss by interpolating between `Init-S` and `Sol-F`, i.e., $\mathcal{L}((1-\alpha)\boldsymbol{\theta}_{\texttt{Init-S}} + \alpha\boldsymbol{\theta}_{\texttt{Sol-F}})$, and present the results in figure 6a. The monotonicity implies that `Sol-F` might be attainable from `Init-S` by gradient-based optimization methods, e.g., SGD, if we are allowed to schedule the training batch in a smart way to recover this linear path. To shed some light on this direction, we visualize the cosine similarity between the negative gradient direction of each example with the direction from `Init-S` to `Sol-F`, *i.e.*, $\boldsymbol{\theta}_{\texttt{Sol-F}} - \boldsymbol{\theta}_{\texttt{Init-S}}$, in figure 6b and figure 6c. We can observe that at the beginning, the cosine similarities are mostly concentrated around 0, which means if we randomly sample a batch of data, the update direction is more likely to be orthogonal to the $\boldsymbol{\theta}_{\texttt{Sol-F}} - \boldsymbol{\theta}_{\texttt{Init-S}}$. This implies the data scheduling will play a more crucial role at the initial phase, i.e., when α is small, to recover the linear path. As α increases, we find that the mean of the cosine similarity is positive, which indicates the optimization enters a contractive region, where the scheduling of data may not be that important.

However, finding the data scheduling may rely on the information of $\boldsymbol{\theta}_{\texttt{Sol-F}} - \boldsymbol{\theta}_{\texttt{Init-S}}$, which is not available for sparse scratch. Therefore, we plan to further analyze the patterns, e.g., the trace of FIM, of the data scheduling that can recover the linear path so as to offer insights on designing criteria for scheduling data without knowing $\boldsymbol{\theta}_{\texttt{Sol-F}} - \boldsymbol{\theta}_{\texttt{Init-S}}$.



(a) (b) (c)

Figure 6: Linear interpolation results on ResNet32 trained on CIFAR100. (a) The linear interpolation path. The red curve shows the test loss for the interpolated network $f((1-\alpha)\boldsymbol{\theta}_{\texttt{Init-S}} + \alpha\boldsymbol{\theta}_{\texttt{Sol-F}})$ with different interpolation coefficients α. While we observe in Figure 2 that `Sol-S` and `Sol-F` has a performance gap, this path shows that we may close the gap by following the linear interpolation path which is monotonically decreasing. (b) & (c) Distribution of the cosine similarity of per sample gradient to the linear interpolation direction from the weights of `Init-S` to `Sol-F`.

Additionally, the work which is mostly aligned with our idea in data schedule might be [57], which shows that it is possible to construct a subset of training dataset in each iteration that consists of two

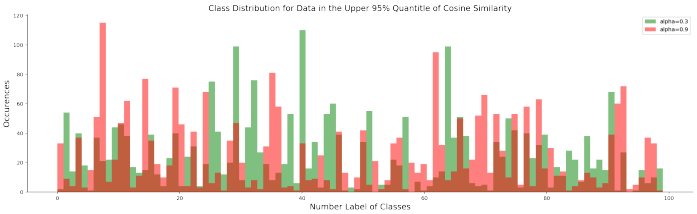


Figure 7: This figure shows the class distribution for the data whose cosine values are of the upper 95% quantile. Clearly, the close-to-init one ($\alpha = 0.3$) and the close-to-solution ($\alpha = 0.9$) one has different class distributions.

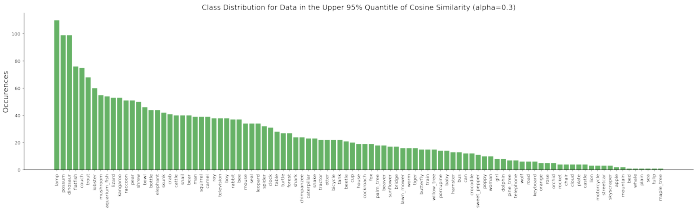


Figure 8: This figure shows a more fine-grained results of the class distribution for the data whose cosine values are of the upper 95% quantile when $\alpha = 0.3$. Some classes disproportionally have higher cosine values, implying that there may exist pattern to schedule data for improving the network performance.

types of data: (1) the data points which are *hard to memorize*, and (2) the data points where the sparse neural networks has *high disagreement with the prediction of the dense neural networks*. However, there is an unavoidable blemish of this approach: the hardest examples inevitably contains some noisy or corrupted samples, such that repetitively reinforcing the memorization on them is harmful. We here propose a simple heuristics based on the critical learning period to fix the issue: we should sample a subset of hardest-to-memorize examples at the beginning to train the sparse neural network (the **memorization phase**), then gradually decrease the average difficulty of the training samples to some certain level (the **consolidating phase**). A promising proxy for the hardness of memorization would be Fisher information.

5 Conclusions and Future Work

In conclusion, we identify and characterize the performance gap between sparse scratch and sparse finetuning by two regimes: the generalization regime and the optimization regime. We provide a unified understanding by analysis in Fisher information and data memorization. For next steps, we consider to propose practical and efficient training schedules to improve the performance of sparse training from scratch, paving the way for truly efficient deep network training.

Remarks

Acknowledgements. We would like to thank Yi Sun for the detailed discussions and helpful feedback on the experimental design for verifying the memorization effect in sparse training. We also thank Aoming Liu for his help in running initial experiments. We acknowledge Huan Wang and Yue Bai's involvement at the initial stage of the project. The project was supported in part by DOE grant DE-EE0009505. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

Author contributions. Y. Chen supported and supervised the project. Z. Ye led the project, proposed the high-level research ideas, scoped the technical content, designed and implemented the experiments, and led the write-up of the paper. C. Wang proposed initial research questions, conducted trial experiments to validate the two regimes, and together with Z. Ye, wrote Sections 1 and 4 and conceived research ideas in closing the performance gap. Z. Ding participated in the research discussions and assisted in the initial draft on related works and the appendices.

References

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

[2] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

[3] Alessandro Achille, Giovanni Paolini, and Stefano Soatto. Where is the information in a deep neural network? *arXiv preprint arXiv:1905.12213*, 2019.

[4] 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.

[5] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, 2021.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Yuan Cao, Zixiang Chen, Mikhail Belkin, and Quanquan Gu. Benign overfitting in two-layer convolutional neural networks. *arXiv preprint arXiv:2202.06526*, 2022.

[8] Shih-Kang Chao, Zhanyu Wang, Yue Xing, and Guang Cheng. Directional pruning of deep neural networks. *Advances in Neural Information Processing Systems*, 33:13986–13998, 2020.

[9] Satrajit Chatterjee. Learning and memorization. In *International Conference on Machine Learning*, pages 755–763. PMLR, 2018.

[10] Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

[11] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkeZisA5t7.

[12] Tianlong Chen, Zhenyu Zhang, pengjun wang, Santosh Balachandra, Haoyu Ma, Zehao Wang, and Zhangyang Wang. Sparsity winning twice: Better robust generalization from more efficient training. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SYuJXrXq8tw.

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[16] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. In *Identifying and Understanding Deep Learning Phenomena Workshop, International Conference on Machine Learning*, 2019.

[17] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[18] 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.

[19] 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.

[20] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[21] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ig-VyQc-MLK.

[22] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OcKMT-36vUs.

[23] Anna Golubeva, Guy Gur-Ari, and Behnam Neyshabur. Are wider nets better given the same number of parameters? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_zx8Oka09eF.

[24] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. *Advances in neural information processing systems*, 31, 2018.

[25] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[26] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019.

[27] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[28] Stanislaw Jastrzebski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *International Conference on Machine Learning*, pages 4772–4784. PMLR, 2021.

[29] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[31] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[32] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[33] 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.

[34] Dawei Li, Tian Ding, and Ruoyu Sun. On the benefit of width for neural networks: Disappearance of bad basins. *arXiv preprint arXiv:1812.11039*, 2018.

[35] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in neural information processing systems*, 33: 20331–20342, 2020.

[36] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[37] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, et al. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? *Advances in Neural Information Processing Systems*, 34, 2021.

[38] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

[39] Jörg Martin and Clemens Elster. Inspecting adversarial examples using the fisher information. *Neurocomputing*, 382:80–86, 2020.

[40] 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):1–12, 2018.

[41] Gergely Neu. Information-theoretic generalization bounds for stochastic gradient descent. In *Conference on Learning Theory*, pages 3526–3545. PMLR, 2021.

[42] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[43] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

[44] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[45] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[46] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ry_WPG-A-.

[47] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[48] Ravid Shwartz-Ziv, Amichai Painsky, and Naftali Tishby. Representation compression and generalization in deep neural networks, 2019. URL https://openreview.net/forum?id=SkeL6sCqK7.

[49] 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.

[50] Darko Stosic and Dusan Stosic. Search spaces for neural model training. *arXiv preprint arXiv:2105.12920*, 2021.

[51] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33: 6377–6389, 2020.

[52] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pages 9636–9647. PMLR, 2020.

[53] 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.

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[55] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Michael C. Mozer, and Yoram Singer. Identity crisis: Memorization and generalization under extreme overparameterization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1l6y0VFPr.

[56] Xiao Zhang, Haoyi Xiong, and Dongrui Wu. Rethink the connections among generalization, memorization and the spectral bias of dnns. *arXiv preprint arXiv:2004.13954*, 2020.

[57] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang. Efficient lottery ticket finding: Less data is more. In *International Conference on Machine Learning*, pages 12380–12390. PMLR, 2021.

[58] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A Experimental Settings

For Figure 1, 6 – 8 and Table 2, we use ResNet32 with widen factor of 4 as the network structure. The default learning rate setting is 0.1, and decay by 0.1 at the 100-th, 150-th, and the 200-th epoch. We tune the learning rate under different sparsity ratio in order to get the best performance. The total training epochs are 250 and we pick the best model with regard to the test performance to present in the figures. The optimizer is SGD with momentum of 0.9 and weight decay of 0.0002. The batch size is 128. For Figure 4 and Figure 5, the major training settings has been specified in the caption, and all the other settings are the same as the above. For Figure 3, we use MLP with ReLU as the activation, with the hidden dimensions being 64-32-32-18-18-6-6. The learning rate is 0.001 and decay by 0.1 at the 500-th, 1000-th and 1500-th epoch, and the total epochs are 1000 for dense training and 2000 for sparse training. We use SGD with zero momentum and weight decay as similarly done in [47]. The batch size is 12.

Our code will be made publicly available after the peer-reviewing phase of the work.

B Class Distribution for Data on which `Sol-S` and `Sol-F` Has Disagreement

Figure 9 and 10 show the class distribution for data on which `Sol-S` and `Sol-F` has disagreement, on the train and test set respectively (they have similar patterns) when sparsity ratio is 0.97. The high-level message is that the the classes are of skewed distributions; or in other words, sparse training from scratch disproportionately impacts a small subset of data (e.g., man, woman, boy, girl).

The skewness of the distribution at least tells us that data schedule could be useful, that some data is most suffered from sparse scratch, so that we may first improve upon that. (The underlying reason may be complex, for example, some data (like the sub-classes of human) may be harder to differentiate from others such that the network must memorize certain low-level features in order to have good performance.

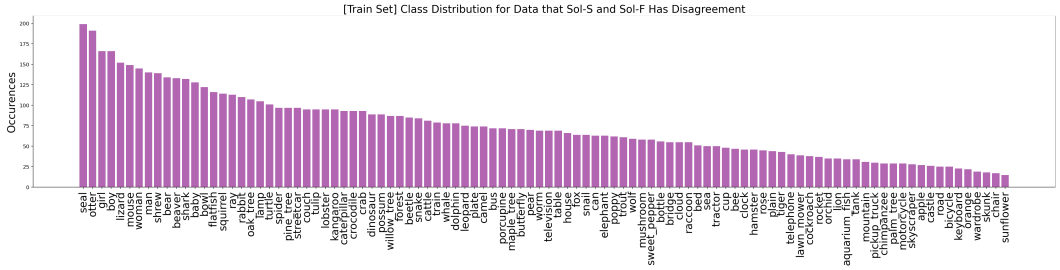


Figure 9: (Train Set) Class distribution for data on which `Sol-S` and `Sol-F` has disagreement. The experimental setting is the same as in Figure 1.

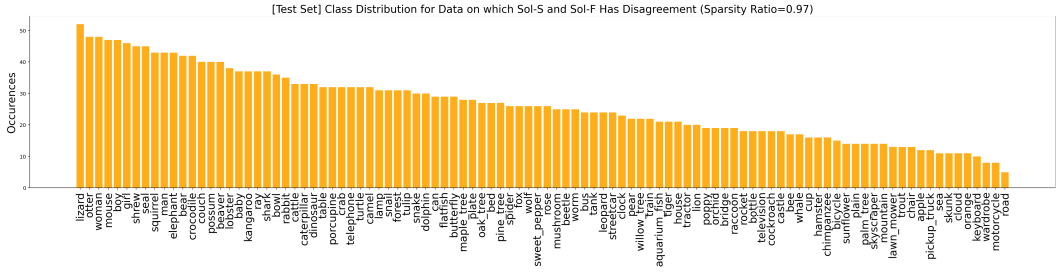


Figure 10: (Test Set) Class distribution for data on which `Sol-S` and `Sol-F` has disagreement. The experimental setting is the same as in Figure 1.

C Additional Trial Experiments on the Signal-to-Noise Ratio of Gradients

We here present additional yet rough and preliminary results in order to archive the work in progress. As mentioned in Section 2, Fisher information can be connected to the (Shannon) Information Bottleneck Theory. Specifically, [47] shows that there exists a fitting phase (where the Shannon mutual information of inputs and layer activations $\mathbf{I}(\mathbf{x}; \mathbf{t}_{\boldsymbol{\theta}})$ increases) and compression phase of network learning (where $\mathbf{I}(\mathbf{x}; \mathbf{t}_{\boldsymbol{\theta}})$ drops), and the compression phase leads to the generalization of deep neural networks. [2, 45] later shows that the phase change in Shannon mutual information during learning corresponds to the change of Fisher information during learning (which also first increase then drops).

Thus, an intuitive interpretation on the trend of Fisher information of `Sol-S` in the two regimes (which both ends with a very high level) implies that `Sol-S` may not be able to compress the data representation well, resulting in its worse generalization performance.

We plan to investigate more on this compression phenomenon. However, Shannon mutual information is notoriously difficult to estimate [45, 11]. Instead, we plot network gradients' signal to noise ratio (SNR, defined from eq. 26 and eq. 27 in [46]) as a proxy, since it has been proved, when training with SGD, that the drifting phase and the diffusion phase of gradient SNR is highly relavant to the fitting and compression phase of Shannon mutual information in deep neural networks (more details of this line of discussions can be found in [1, 48, 45, 10]).

Our rough and preliminary toy results are presented in Figure 11, which vaguely shows some connections of gradient SNR patterns to the two regimes. We plan to conduct more rigorous experiments in future revisions.

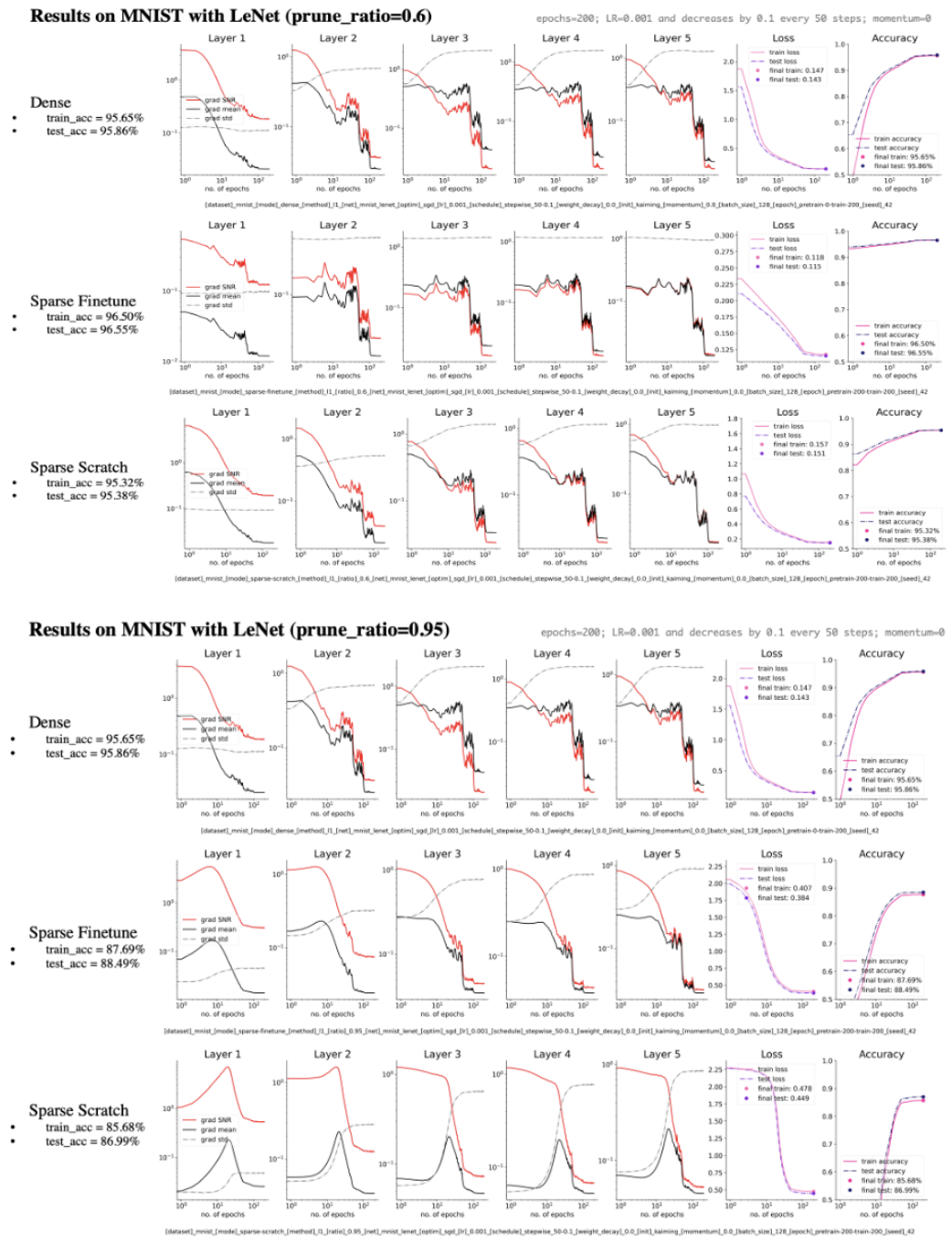


Figure 11: **Trend of Gradient SNR.** We plot layer-wise evolution of gradients of stochastic batches for three training methods: sparse training from scratch, sparse finetuning and dense training. Both plots are trained on MNIST with LeNet with learning rate $= 0.001$ and decreases by 0.1 every 50 steps for total of 200 epochs. The left plot has 60% prune ratio and roughtly represents the generalization regime (notice this is inaccurate since the training performance is not near-optimal) and the right plot has 95% prune ratio and represents the optimization regime. In generalization regime, the standard deviation of the gradients of the sparse training from scratch method is slightly lower than sparse finetuning and dense neural networks, which indicates lack of diffusion in the later SGD dynamics thus worse generalization capability. The right plot indicates sparse training from scratch starts with very low gradient mean and there exists a hike during the trajectory; we believe the hike is highly relevant to the optimization failure in the optimization regime.