Generalization and Memorization in Sparse Neural Networks

Ziyu Ye ZIYUYE@UCHICAGO.EDU
Department of Computer Science
The University of Chicago

Chaoqi Wang CHAOQI@UCHICAGO.EDU
Department of Computer Science
The University of Chicago

Yuxin Chen CHENYUXIN@UCHICAGO.EDU
Department of Computer Science
The University of Chicago

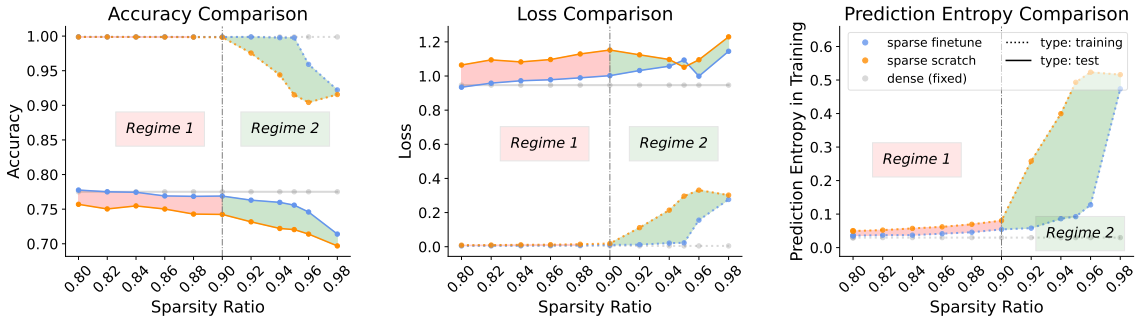
Abstract

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 require more information in learning, and are weaker at memorization*; 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 (Devlin et al., 2018; Brown et al., 2020; Chowdhery et al., 2022). 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 (Patterson et al., 2021). 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 (Hoefler et al., 2021).

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 initializing parameters from scratch (Lee et al., 2019; Wang et al., 2020; Tanaka et al., 2020), 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. That is, the solution of sparse scratch (*i.e.*, `Sol-S`) has significantly



(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 scratch (`Sol-S`, orange line) and sparse finetuning (`Sol-F`, blue line) on CIFAR-100 with ResNet32 on varying sparsity ratios. Experimental specifications and detailed results with standard deviations reported are 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 denote 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 denote this regime as the *optimization regime*. (b) **Loss**. Similar patterns exist in regime 1 and regime 2. (c) **Training prediction entropy**. `Sol-S` has a higher prediction entropy and the situation is exceptionally noticeable in regime 2, implying `Sol-S` is much less confident in its prediction.

worse *test accuracy* than that of the solution of sparse finetuning (*i.e.*, `Sol-F`). Considering the remarkable benefits of sparse scratch, we ask:

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

Though there are some initial attempts (Evci et al., 2019; Stosic and Stosic, 2021; Frankle et al., 2021) to understand this performance gap, it is still unclear what the fundamental mechanism is underlying, and how to improve sparse scratch 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 closing the performance gap. We find that there exist two regimes as shown in Figure 1, where sparse scratch cannot generalize as well as sparse finetuning, 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 in regime 2, *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. Our key contributions are summarized 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.
- With extensive experiments, we offer a consistent explanation for the two regimes: *sparse scratch requires more information in learning, and is weaker at memorization.* We identify the key factors for the worse performance of sparse scratch in the generalization regime (*e.g.*, sharper local minima and higher sensitivity) and the optimization regime (*e.g.*, 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.

1. While the two regimes here may be naïvely separated by the sparsity ratio, there may exist more nuances (*e.g.*, settings for learning rate and batch size) to be discovered by future works.

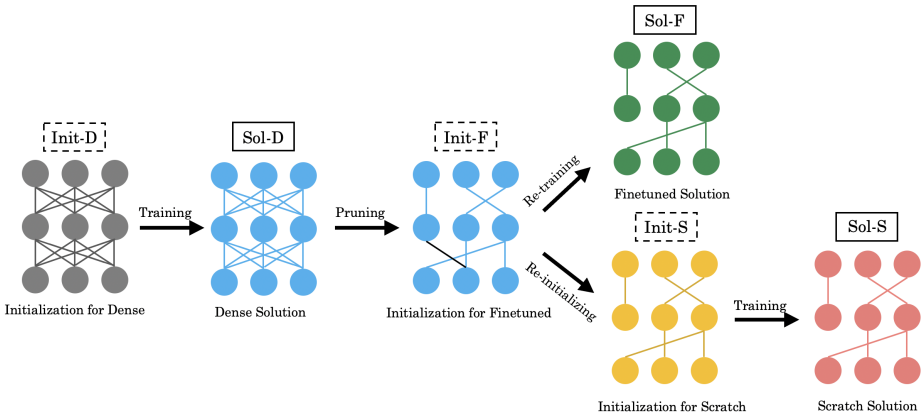


Figure 2: **Training procedures.** We start with a densely connected network whose parameters are denoted as `Init-D` and the trained **dense solution** is denoted as `Sol-D`. To find a sparse mask, we prune `Sol-D` and the remaining parameters are denoted as `Init-F`. We consider two types of sparse 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 (Evci et al., 2019).

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.

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 (Zhu and Gupta, 2017).[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 (LeCun et al., 1989). Though being simple, this strategy has shown to be a solid pruning strategy (Liu et al., 2018; Evci et al., 2019), 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 (Dinh et al., 2017; Tsuzuku et al., 2020; Gilmer et al., 2022). 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 (Keskar et al., 2017; Neyshabur et al., 2017).

2. Many related works also discuss the lottery setting (Frankle and Carbin, 2019; Evci et al., 2019) (different from the sparse training from scratch, it re-uses subset of the parameters from `Init-D` as the initialization) and multi-shot or dynamic sparse training strategies (Mocanu et al., 2018; Evci et al., 2020; Ma et al., 2021). We plan to incorporate comparisons with those settings in the future revisions.

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 captures the local sensitivity of the network to the input. This quantity[3] is shown to be well correlated with the generalization and robustness of deep neural networks (Novak et al., 2018).

Memorization in deep neural networks. Recent works have empirically demonstrated the impact of *label memorization* on the generalizability or trainability of neural networks (Arpit et al., 2017; Chatterjee, 2018; Zhang et al., 2020b; Stephenson et al., 2021). 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.) (Feldman, 2020; Feldman and Zhang, 2020; Brown et al., 2021). This phenomenon may also be referred to as "benign overfitting" (Cao et al., 2022), and different architectures have shown to possess different inductive bias towards memorization (Zhang et al., 2020a).

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 Martens (2014); Achille et al. (2019), $\mathbf{F}(\boldsymbol{\theta})$ 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), we have $\mathbf{H}(\boldsymbol{\theta}) \approx \mathbf{F}(\boldsymbol{\theta})$ to the first-order approximation.
- **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

  $$\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})} \mathrm{KL}\left(p_{\boldsymbol{\theta}'}(y \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(y \mid \mathbf{x})\right) = \delta\boldsymbol{\theta}^\top \mathbf{F} \delta\boldsymbol{\theta} + o\left(\|\delta\boldsymbol{\theta}\|_2^2\right). \tag{2.4}$$

  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 (Achille et al., 2018; Kirkpatrick et al., 2017; Achille et al., 2019).
- **Relationship to data memorization**: Some initial attempts show that Fisher information is relevant to data memorization, which in turn impacts the networks' generalization ability (Achille et al., 2018, 2019; Martin and Elster, 2020; Jastrzebski et al., 2021). 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.

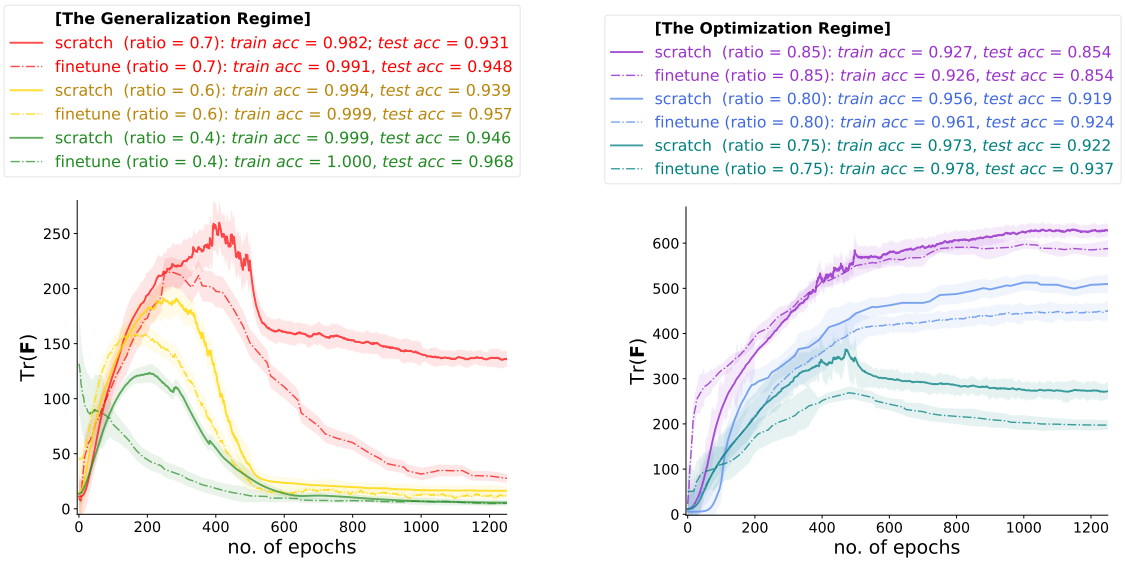


Figure 3: **The training dynamics of trace of Fisher information.** The experiments use a synthetic dataset for binary classification with fully connected networks.[6] The plots shows (1) a clear distinction between the generalization regime (*left*) and the optimization regime (*right*), as well as (2) the difference between sparse finetuning (*thick lines*) and sparse scratch (*dashed lines*).

	Training Error	Trend of Fisher Information	Reasons for the Performance Gap
Generalization Regime	near optimal	increases first, then decreases, yet ending up higher than finetuning's	sharper minima, higher sensitivity
Optimization Regime	suboptimal	increases and hardly decreases	weaker memorization on training data

Table 1: Summary of the correlation of the two regimes with the dynamics of Fisher information during training, and the major reasons for the performance gap on sparse scratch.

3. Experimental Results

Learning to generalize is like **crossing a barrier** or break through a bottleneck.

By Corollary A.1, the high Fisher information implies that even a small perturbation to the parameter can bring large discrepancy of the network prediction. Empirically, this typically happens when the network is (1) *learning new concepts* (*cf.* Shwartz-Ziv and Tishby (2017); Achille et al. (2019)), or (2) *memorizing hard examples* (*cf.* Martin and Elster (2020); Jastrzebski et al. (2021)).

Intuitively, for sparse scratch, those new concepts (yet which may be familiar to sparse finetuning given the pretraining) and hard examples (which may contain both *worth-learning data* with subtle features and *not-worth-learning data* with noisy labels) incur much higher Fisher information.

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.
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 (Shwartz-Ziv and Tishby, 2017; Achille and Soatto, 2018; Chaudhari and Soatto, 2018). We provide additional experiments and analysis on this in the appendix.
6. Check Appendix B for experimental details, and see Appendix C for similar results on CIFAR100 with ResNet32.

- In the generalization regime, while sparse scratch can manage to fit all those data, the higher fisher information coincides with tortured loss landscapes (*e.g.*, sharper local minima and higher sensitivity), leading to worse performance.
- In the optimization regime, sparse scratch can not even fit some of those hard examples, resulting in drop in both training and test errors.

TODO: add corr plots on [disagreement v.s. Fisher information] w.r.t. class to confirm the intuition.

TODO: Also point to the appendix with formal review on measurable information of neural network (e.g., the number of bits needed to encode the weights given certain training performance, which is closely related to fisher information).

TODO: 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.

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**.

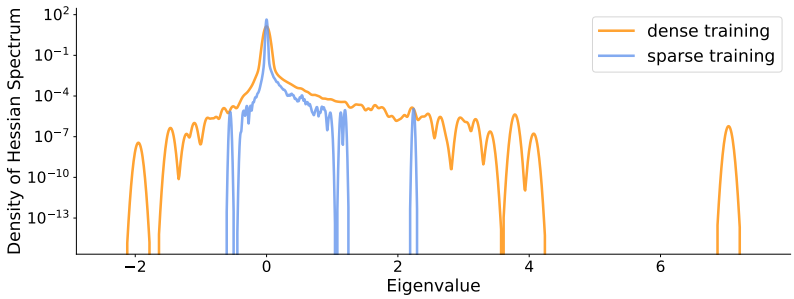


Figure 4: **The distribution of Hessian spectrum.** This plot shows the distribution of the spectrum for the loss Hessian at the training convergence for sparse scratch (`Sol-S`, orange line) and sparse finetuning (`Sol-F`, blue line) on CIFAR-100 with ResNet 32 at a sparsity ratio of 0.90 (*i.e.*, at the generalization regime) (To update the figure to match the ratio).

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 (Arpit et al., 2017; Golubeva et al., 2021), 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).

Sparsity Ratio	Tr(F)		Tr(H)		Jacobian Norm		Entropy	
	Sol-F	Sol-S	Sol-F	Sol-S	Sol-F	Sol-S	Sol-F	Sol-S
0.80	218.29	682.79 ↑	131.66	353.87 ↑	32.38	45.20 ↑	0.03	0.05 ↑
0.82	242.64	990.51 ↑	124.05	452.80 ↑	33.56	49.50 ↑	0.03	0.05 ↑
0.84	295.69	984.73 ↑	166.05	452.74 ↑	34.25	48.19 ↑	0.03	0.05 ↑
0.86	351.47	1329.26 ↑	175.58	484.45 ↑	36.03	53.66 ↑	0.04	0.06 ↑
0.88	454.83	1901.46 ↑	172.85	596.07 ↑	37.98	47.39 ↑	0.04	0.07 ↑
0.90	635.20	2898.05 ↑	292.47	722.28 ↑	41.22	57.26 ↑	0.05	0.08 ↑

Table 2: **Fisher information, loss curvature, sensitivity and entropy.** The experimental setup of this table is the same as in Figure 1 on CIFAR100 with ResNet32. The entropy refers to the prediction entropy evaluated on the training set. Estimation methods can be found in Appendix B, and the full table with standard deviation reported is in Appendix **??**.

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.*

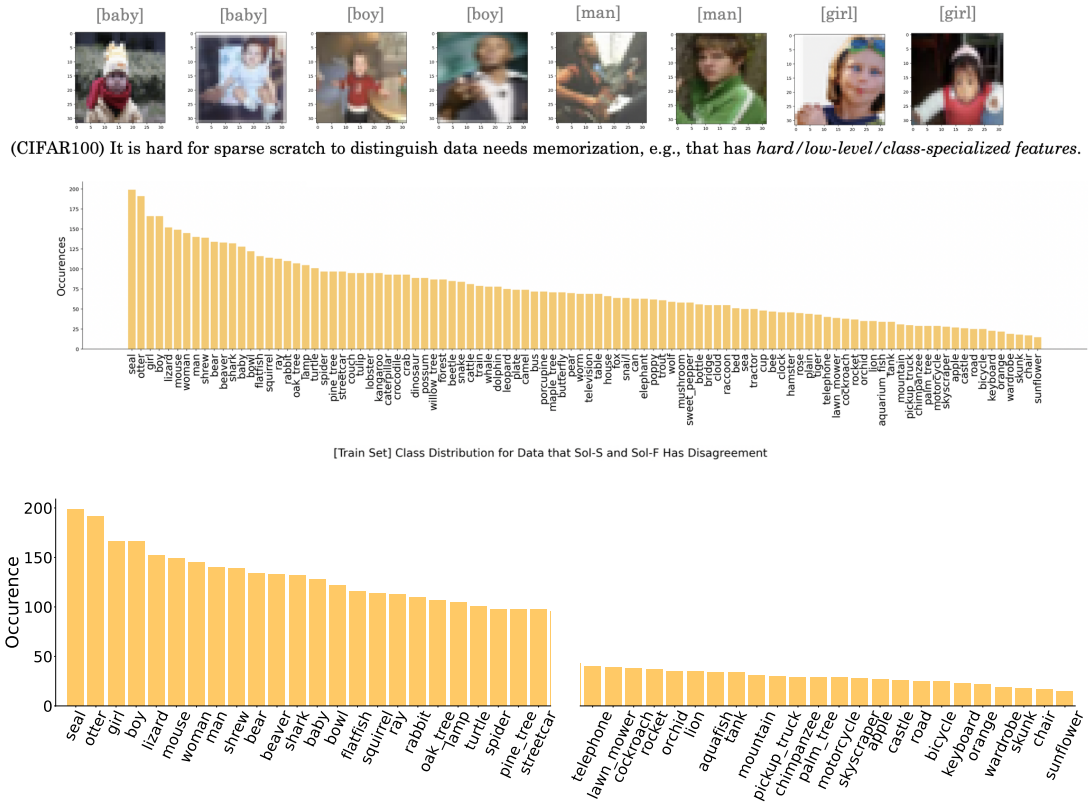
To explain, since the initialization of sparse finetuning inherits weights from a well-trained densely connected network, it may already entail some memorization on data labels (especially for those hard examples). However, sparse scratch initializes its weights randomly, and since memorization happens only at the late phase of training (Liu et al., 2020), 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 requires memorization (which is computationally expensive (Koh and Liang, 2017; Feldman and Zhang, 2020)),[7] we adopt a noisy training setting, and shows that `Sol-S` memorize less on those noisy data, as demonstrated in Figure 6.

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.[8] Our results may also shed lights on the barrier between `Sol-F` and `Sol-T` as discussed in (Li et al., 2018; Evci et al., 2019).

Ziyu: To calculate the fisher information of each example and sort by that, and compare with the disagreement plot.

7. As a side note, Hooker et al. (2019) 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`.
8. Some recent work also discuss the robustness of sparsity in the adversarial setting (Chen et al., 2022; Guo et al., 2018).



(CIFAR100) It is hard for sparse scratch to distinguish data needs memorization, e.g., that has *hard/low-level/class-specialized features*.

Figure 5: **Sparse scratch memorizes worse than sparse finetuning.** It is currently a placeholder illustrating what we will do here. The plots are hard to read now.

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 Chao et al. (2020) 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 Jastrzebski et al. (2021) which proposes a loss regularizer with respect to Fisher information. However, they apply 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.

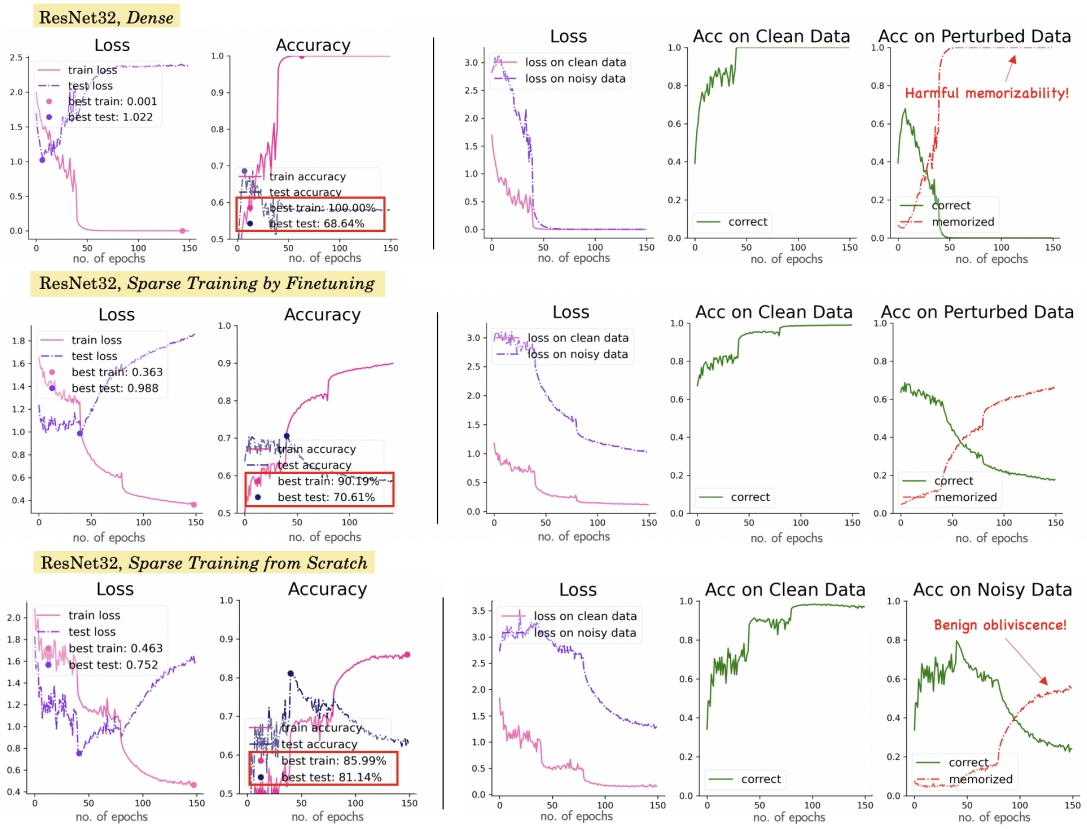


Figure 6: **Sparse training from scratch is more robust to label noise.** This experiment is conducted on CIFAR-10 with ResNet32. The training set contains 30% perturbed data whose labels are uniformly randomly shuffled as similarly conducted in (Liu et al., 2020). The sparsity ratio is 0.95 for sparse 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.

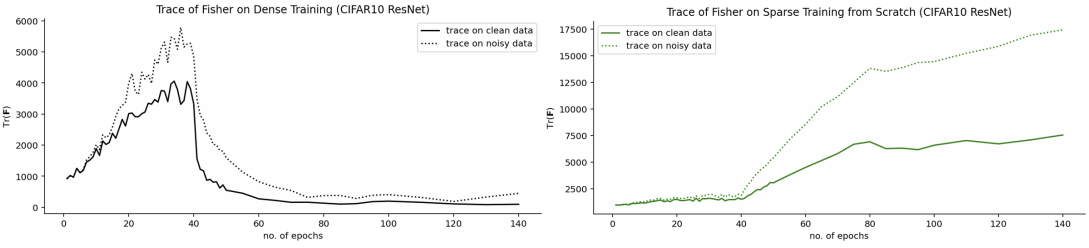


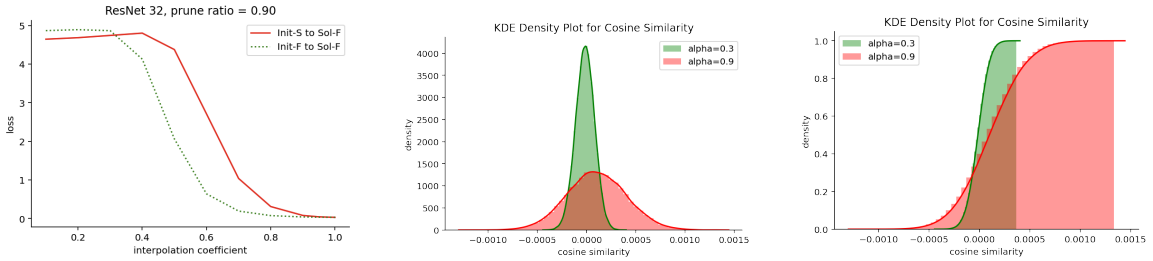
Figure 7: **Trend of Fisher information.** The experimental setup is the same as in Figure 6. As a supplementary figure, the results demonstrate that the Fisher information is higher for those noisy data, and sparse scratch has a higher Fisher information which hardly decrease when the network can not memorize all the training data.

4.2 The Data Schedule Perspective

4.2.1 Insights from the Linear Interpolation Path

In Evci et al. (2019), 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 8a. 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 8b and figure 8c. 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 8: 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 Zhang et al. (2021), which shows that it is possible to construct a subset of training dataset in each iteration that consists of two 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.

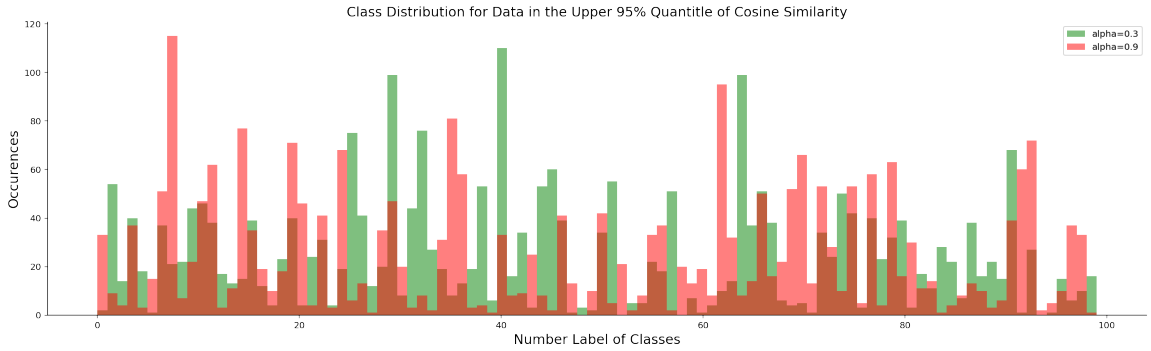


Figure 9: This figure shows the class distribution for the data whose cosine values are of the upper 95% quantile. Clearly, the close-to-init one ($\alpha = 0.3$) and the close-to-solution ($\alpha = 0.9$) one has different class distributions.

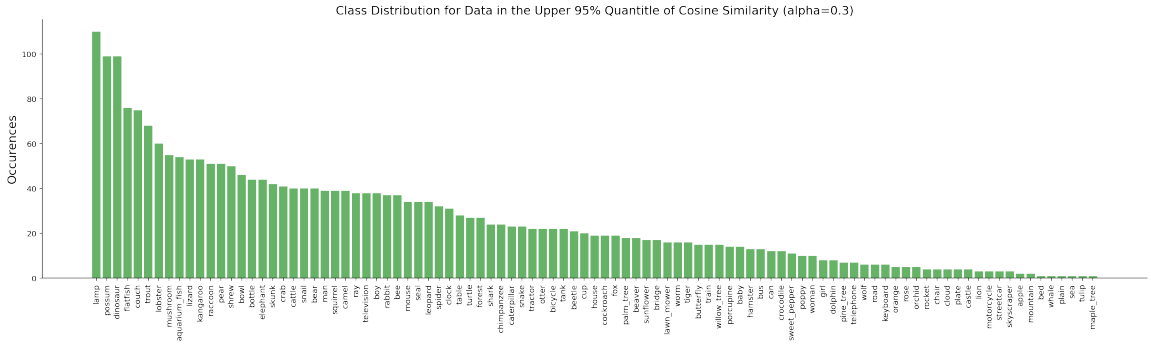


Figure 10: This figure shows a more fine-grained results of the class distribution for the data whose cosine values are of the upper 95% quantile when $\alpha = 0.3$. Some classes disproportionally have higher cosine values, implying that there may exist pattern to schedule data for improving the network performance.

4.2.2 Data Scheduling by Fisher Information

Note: to add the bi-level algorithm, that is to first to select by loss (meaning using label information), then select from that subset by fisher information (no label information, meaning that select the points which change the distribution most).

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.

Algorithm 1 A Toy Data Curriculum by Fisher Information

Input: train set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, batch size $n_\mathcal{B}$, pool size $n_\mathcal{P} > n_\mathcal{B}$, step size η, phase time $t^\star$

Initialize weight parameter $\boldsymbol{\theta}^t$ where the time step $t = 0$
▽ /* Fisher information guided SGD phase* /
for $t = 1, \ldots, t^\star$ **do**
  Uniformly randomly select a large input data pool $\mathcal{P}^t$ of size $n_\mathcal{P}$ from $\mathcal{D}$
  $\forall\ \boldsymbol{x}_i \in \mathcal{P}^t$, compute $\texttt{Fisher}[\boldsymbol{\theta}^t \mid \boldsymbol{x}_i]$, the (trace of) Fisher information of $\boldsymbol{\theta}^t$ given $\boldsymbol{x}_i$
  $\mathcal{B}^t \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_\mathcal{B}}$, the **bottom-**$n_\mathcal{B}$ samples in $\mathcal{P}^t$ in terms of the criteria $\texttt{Fisher}$
  $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$
end for
▽ /* Regular SGD phase * /
for $t = t^\star + 1, \ldots, T$ **do**
  Uniformly randomly select a data batch $\mathcal{B}^t$ of size $n_\mathcal{B}$ from $\mathcal{D}$
  $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$
end for

Algorithm 2 (Baseline) Fisher Information Selection

Input: Training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, batch size $n_\mathcal{B}$, pool size $n_\mathcal{P} > n_\mathcal{B}$, learning rate η

Initialize weight parameter $\boldsymbol{\theta}^t$ where the time step $t = 0$
for $t = 1, \ldots, T$ **do**
  ▽ /* initialize a pool and compute the criteria * /
  Uniformly randomly select a large input data pool $\mathcal{P}^t$ of size $n_\mathcal{P}$ from $\mathcal{D}$
  $\forall\ \boldsymbol{x}_i \in \mathcal{P}^t$, compute $\texttt{Fisher}[\boldsymbol{\theta}^t \mid \boldsymbol{x}_i]$, the (trace of) Fisher information of $\boldsymbol{\theta}^t$ given $\boldsymbol{x}_i$
  ▽ /* create the subset by sorting and selection * /
  $\mathcal{B}^t \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_\mathcal{B}}$, the top-$n_\mathcal{B}$ samples in $\mathcal{P}^t$ in terms of the criteria $\texttt{Fisher}$
  ▽ /* update parameter * /
  $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$ where the mini-batch gradient is $\nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$
end for

Algorithm 3 Reducible Fisher Information Selection

Input: Training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, batch size $n_\mathcal{B}$, pool size $n_\mathcal{P} > n_\mathcal{B}$, learning rate η
Input: Parameter $\boldsymbol{\theta}^{\text{ho}}$ from a small model trained on a small holdout set $\mathcal{D}^{\text{ho}}$

$\forall\ \boldsymbol{x}_i \in \mathcal{D}$, compute $\texttt{Fisher}[\boldsymbol{\theta}^{\text{ho}} \mid \boldsymbol{x}_i]$ ▷ /* compute the holdout Fisher information * /
Initialize weight parameter $\boldsymbol{\theta}^t$ where the time step $t = 0$
for $t = 1, \ldots$ **do**
  ▽ /* initialize a pool and compute the criteria * /
  Uniformly randomly select a large input data pool $\mathcal{P}^t$ of size $n_\mathcal{P}$ from $\mathcal{D} \setminus \mathcal{D}^{\text{ho}}$
  $\forall\ \boldsymbol{x}_i \in \mathcal{P}^t$, $\texttt{ReduceFisher}[\boldsymbol{x}_i] \leftarrow \texttt{Fisher}[\boldsymbol{\theta}^t \mid \boldsymbol{x}_i] - \texttt{Fisher}[\boldsymbol{\theta}^{\text{ho}} \mid \boldsymbol{x}_i]$
  ▽ /* create the subset by sorting and selection * /
  $\mathcal{B}^t \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_\mathcal{B}}$, the top-$n_\mathcal{B}$ samples in $\mathcal{P}^t$ in terms of the criteria $\texttt{ReducibleFisher}$
  ▽ /* update parameter * /
  $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$ where the mini-batch gradient is $\nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}(\boldsymbol{\theta})$
end for

Acknowledgments and Disclosure of Funding

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 thank Aoming Liu, Huan Wang, Yue Bai and Zixin Ding for their constructive discussions. 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.

References

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

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

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

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.

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.

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

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.

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

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.

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.

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.

Harald Cramér. *Mathematical Methods of Statistics*, volume 43. Princeton University Press, 1946.

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.

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.

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.

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.

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.

Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222 (594-604):309–368, 1922.

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.

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.

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.

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.

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

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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.

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

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.

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.

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.

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

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

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

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

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.

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.

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.

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

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.

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

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

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.

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

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.

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.

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

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.

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

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.

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.

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.

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*, 2020a. URL https://openreview.net/forum?id=B1l6y0VFPr.

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

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.

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. Notes on Information in Deep Neural Networks

As a supplement to Section 3, we here provide a detailed discussion on information in deep neural networks. Consider a classification task with a conditional distribution $p(\mathrm{y}|\mathbf{x})$, where the random variable $\mathbf{x} \sim p(\mathbf{x})$ denotes an input (*e.g.*, an image) and the random variable $\mathrm{y} \in \mathbb{Y} = \{1, \cdots, C\}$ denotes the target. Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be a training dataset *i.i.d.* sampled from $p(\mathbf{x}, \mathrm{y}) = p(\mathbf{x})p(\mathrm{y}|\mathbf{x})$. A neural network $f(\cdot; \boldsymbol{\theta})$ with the weight parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ encodes a conditional distribution $q_{\boldsymbol{\theta}}(\mathrm{y}|\mathbf{x})$ by fitting the dataset. We denote the network prediction for $\boldsymbol{x}_i$ as $\hat{y}_i$.

The objective is to minimize the negative log likelihood loss (*i.e.*, the cross-entropy loss) on $\mathcal{D}$:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \ell(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) \tag{A.1}$$

$$= \frac{1}{N} \sum_{i=1}^{N} -\log q_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i). \tag{A.2}$$

We take the mini-batch stochastic gradient descent (SGD) for the minimization. Let $\mathcal{B} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_{\mathcal{B}}}$ denotes a random mini batch *i.i.d.* sampled from $\mathcal{D}$. The average loss gradient of the random mini-batch is defined as $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{n_{\mathcal{B}}} \sum_{\boldsymbol{x}_i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}_i, y_i; \boldsymbol{\theta})$. Given a learning rate η and time step t, we update the weight parameter by:

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}). \tag{A.3}$$

We introduce **KL divergence** (also termed as relative entropy) as a useful notion for the following discussion; it can be seen as an intrinsic dissimilarity metric for distributions (Kullback and Leibler, 1951). For any two distributions $q_{\boldsymbol{\theta}}(\cdot)$ and $q_{\boldsymbol{\theta}'}(\cdot)$, the KL divergence between them is:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\cdot) \| q_{\boldsymbol{\theta}'}(\cdot)) \triangleq \mathbb{E}_{q_{\boldsymbol{\theta}}(\cdot)} \left[\log \frac{q_{\boldsymbol{\theta}}(\cdot)}{q_{\boldsymbol{\theta}'}(\cdot)} \right]. \tag{A.4}$$

A.1 Fisher Information

In this sub-section, we first introduce Fisher information from its original parameter estimation view (Fisher, 1922). However, we note that this view is unsuitable in the context of deep learning, thus we discuss an alternative view by taking Fisher information as a **local distance metric for distributions**. This will foster the interpretation of Fisher information as a **measure for information retained by the neural network**, and lay a good foundation for understanding the role of Fisher information in the optimization dynamics and generalization.

Definition A.1. (**Fisher information — an estimation view**). *Imagine $\boldsymbol{\theta}$ as an unknown parameter modeling a distribution $q_{\boldsymbol{\theta}}(\cdot)$. Assuming the log likelihood function $\log q_{\boldsymbol{\theta}}(\cdot)$ is differentiable, we define the* **score function** *of $q_{\boldsymbol{\theta}}(\cdot)$ as the gradient of the log likelihood function $\log q_{\boldsymbol{\theta}}(\cdot)$:*

$$s(\boldsymbol{\theta}) \triangleq \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\cdot). \tag{A.5}$$

The **Fisher information** *of $q_{\boldsymbol{\theta}}(\cdot)$ is defined as the covariance of the score function:*

$$\mathbf{F}(\boldsymbol{\theta}) \triangleq \mathrm{Cov}_{q_{\boldsymbol{\theta}}(\cdot)} \left[s(\boldsymbol{\theta}) \right] \tag{A.6}$$

$$\overset{(a)}{=} \mathbb{E}_{q_{\boldsymbol{\theta}}(\cdot)} \left[(\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\cdot))(\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\cdot))^\top \right], \tag{A.7}$$

where (a) follows from the fact that $\mathbb{E}_{q_{\boldsymbol{\theta}}(\cdot)} \left[s(\boldsymbol{\theta}) \right] = 0$.

Remark A.1. *For neural network parameterized by* $\boldsymbol{\theta}$, $\mathbf{F}(\boldsymbol{\theta})$ *is the* fitted gradient covariance *of the negative log likelihood loss* $\ell(\boldsymbol{x}_i, \hat{y}_i; \boldsymbol{\theta})$, *that is* $\mathbf{F}(\boldsymbol{\theta}) = \mathrm{Cov}_{\boldsymbol{x}_i \sim \mathcal{D}, \hat{y}_i \sim q_{\boldsymbol{\theta}}(\mathrm{y}|\mathbf{x})} \left[\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\hat{y}_i | \boldsymbol{x}_i)\right]$.

As a side note, it is important not to confuse this with the training gradient covariance, *i.e.*, the **empirical Fisher information**, in which $\hat{y}_i$ above is replaced with the true target $y_i \sim p(\mathrm{y}|\mathbf{x})$. The emprical Fisher information may be used to approximate the Fisher information yet it has some limitations (Kunstner et al., 2019), which are not of interests of discussions here.

This classical view is concerned with the parameter estimation problem, and $\boldsymbol{\theta}$ is treated as an *unknown parameter* characterizing a distribution $q_{\boldsymbol{\theta}}(\cdot)$. A high $\mathbf{F}(\boldsymbol{\theta})$ implies that the likelihood function $q_{\boldsymbol{\theta}}(\cdot)$ is rapidly varying,[1] *i.e.*, the likelihood is easily affected by the choice (or value) of $\boldsymbol{\theta}$; thus it is easy to infer the true value of $\boldsymbol{\theta}$ by sampling data from $q_{\boldsymbol{\theta}}(\cdot)$; in other words, the samples can can provide high amount of information to estimate the unknown parameter $\boldsymbol{\theta}$.[2]

However, the estimation view is fundamentally improper to analyze neural networks. In the context of deep learning, $\boldsymbol{\theta}$ is rather a *known parameter* (*i.e.*, network weights) controling the prediction distribution. It is meaningless to estimate $\boldsymbol{\theta}$ through sampling from predictions. The point of focus is not the information that samples from predictions can provide about $\boldsymbol{\theta}$, but rather the information that $\boldsymbol{\theta}$ can provide about the neural network learning process (little literature has explicitly noted this slight distinction, but it can bring potential confusions).

Learning is a dynamic process. The weight parameter $\boldsymbol{\theta}$ is ever changing (updating) with SGD; it is natural to think of the information of $\boldsymbol{\theta}$ from a variation perspective.[3] Intuitively:

> *If small variation in* $\boldsymbol{\theta}$ *results in large discrepancy to the network prediction distribution* $q_{\boldsymbol{\theta}}(\cdot)$, *this* $\boldsymbol{\theta}$ *can be seen as to withhold high amount of information about the learning process.*

We hereby introduce Fisher information as a local metric for such distribution discrepancy.

Lemma A.1. *Assume that the log likelihood function* $\log q_{\boldsymbol{\theta}}(\cdot)$ *is twice differentiable, and denote its Hessian with respect to* $\boldsymbol{\theta}$ *as* $\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\cdot)}$. *The Fisher information of* $q_{\boldsymbol{\theta}}(\cdot)$ *equals the negative of the expected Hessian (i.e., second derivative) of the log likelihood function, that is:*

$$\mathbf{F}(\boldsymbol{\theta}) = -\mathbb{E}_{q_{\boldsymbol{\theta}}(\cdot)} \left[\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\cdot)}\right]. \tag{A.8}$$

Proof. Let's instantiate the likelihood function by $q_{\theta}(\mathbf{z})$, where $\mathbf{z} \sim q_{\theta}(\mathbf{z})$.[4] Then,

$$\begin{aligned}
\mathbf{H}_{\log q_{\theta}(\mathbf{z})} &\triangleq \nabla_{\theta}^2 \log q_{\theta}(\mathbf{z}) \\
&= \nabla_{\theta} \frac{\nabla_{\theta} q_{\theta}(\mathbf{z})}{q_{\theta}(\mathbf{z})} \\
&\overset{(a)}{=} \frac{q_{\theta}(\mathbf{z}) \mathbf{H}_{q_{\theta}(\mathbf{z})} - \nabla_{\theta} q_{\theta}(\mathbf{z}) \nabla_{\theta} q_{\theta}(\mathbf{z})^{\top}}{q_{\theta}(\mathbf{z}) q_{\theta}(\mathbf{z})} \\
&= \frac{\mathbf{H}_{q_{\theta}(\mathbf{z})}}{q_{\theta}(\mathbf{z})} - \left(\frac{\nabla_{\theta} q_{\theta}(\mathbf{z})}{q_{\theta}(\mathbf{z})}\right) \left(\frac{\nabla_{\theta} q_{\theta}(\mathbf{z})}{q_{\theta}(\mathbf{z})}\right)^{\top},
\end{aligned} \tag{A.9}$$

where (a) follows from the quotient rule and $\mathbf{H}_{q_{\theta}(\mathbf{z})} \triangleq \nabla_{\theta}^2 q_{\theta}(\mathbf{z})$.

1. In this note, $q_{\boldsymbol{\theta}}(\cdot)$ may denote a *distribution* or a *likelihood function*. We will add clarifications when necessary.
2. Plus, Cramér–Rao Bound (Cramér, 1946) shows the precision of any unbiased estimator for $\boldsymbol{\theta}$ is at most $\mathbf{F}(\boldsymbol{\theta})$.
3. Notice how this differs from the common definition of information by *Shannon entropy*, which considers information about the *static* distribution per se, instead of the *change* of the distribution by the local parameter variation which we are discussing. An important prior work of this is Achille et al. (2019).
4. The random variable $\mathbf{z}$ can be viewed as a imprecise shorthand notation for $\mathrm{y}|\mathbf{x}$ in our context.

Taking expectation on Eq. A.9:

$$\begin{aligned}
\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\mathbf{z})}\right] &= \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\frac{\mathbf{H}_{q_{\boldsymbol{\theta}}(\mathbf{z})}}{q_{\boldsymbol{\theta}}(\mathbf{z})} - \left(\frac{\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\theta}}(\mathbf{z})}\right)\left(\frac{\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\theta}}(\mathbf{z})}\right)^{\top}\right] \\
&= \int \frac{\mathbf{H}_{q_{\boldsymbol{\theta}}(\mathbf{z})}}{q_{\boldsymbol{\theta}}(\mathbf{z})} q_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z} - \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\left(\frac{\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\theta}}(\mathbf{z})}\right)\left(\frac{\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})}{q_{\boldsymbol{\theta}}(\mathbf{z})}\right)^{\top}\right] \\
&= \int \nabla^2_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z} - \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})^{\top}\right] \\
&\overset{(b)}{=} \nabla^2_{\boldsymbol{\theta}} \int q_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z} - \mathbf{F}(\boldsymbol{\theta}) \\
&= -\mathbf{F}(\boldsymbol{\theta}),
\end{aligned} \tag{A.10}$$

where (b) follows from Definition A.1. Thus $\mathbf{F}(\boldsymbol{\theta}) = -\mathbb{E}_{q_{\boldsymbol{\theta}}(\cdot)}\left[\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\cdot)}\right]$. □

Corollary A.1. (**Fisher information — a metric view**). *Given two distributions $q_{\boldsymbol{\theta}}(\cdot)$ and $q_{\boldsymbol{\theta}'}(\cdot)$ parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ respectively and assuming their likelihood functions are twice differentiable, we have the Fisher information of $\boldsymbol{\theta}$ as the following:*

$$\mathbf{F}(\boldsymbol{\theta}) = \nabla^2_{\boldsymbol{\theta}'} D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\cdot) \| q_{\boldsymbol{\theta}'}(\cdot)\right)\big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}}. \tag{A.11}$$

Proof. By definition in Eq. A.4 and instantiating the likelihood function with the random variable $\mathbf{z}$, the gradient of the KL divergence can be decomposed as:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}'} D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\mathbf{z}) \| q_{\boldsymbol{\theta}'}(\mathbf{z})\right) &= \nabla_{\boldsymbol{\theta}'} \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\log q_{\boldsymbol{\theta}}(\mathbf{z})\right] - \nabla_{\boldsymbol{\theta}'} \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\log q_{\boldsymbol{\theta}'}(\mathbf{z})\right] \\
&= -\nabla_{\boldsymbol{\theta}'} \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\log q_{\boldsymbol{\theta}'}(\mathbf{z})\right] \\
&= -\int q_{\boldsymbol{\theta}}(\mathbf{z}) \nabla_{\boldsymbol{\theta}'} \log q_{\boldsymbol{\theta}'}(\mathbf{z})\mathrm{d}\mathbf{z}.
\end{aligned}$$

Thus, the second derivative of the KL divergence with regard to $\boldsymbol{\theta}'$ evaluated at $\boldsymbol{\theta}' = \boldsymbol{\theta}$ is:

$$\begin{aligned}
\nabla^2_{\boldsymbol{\theta}'} D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\mathbf{z}) \| q_{\boldsymbol{\theta}'}(\mathbf{z})\right)\big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}} &= -\int q_{\boldsymbol{\theta}}(\mathbf{z}) \nabla^2_{\boldsymbol{\theta}'} \log q_{\boldsymbol{\theta}'}(\mathbf{z})\Big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}} \mathrm{d}\mathbf{z} \\
&= -\int q_{\boldsymbol{\theta}}(\mathbf{z}) \mathbf{H}_{\log q_{\boldsymbol{\theta}}(\mathbf{z})} \mathrm{d}\mathbf{z} \\
&= -\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\mathbf{z})}\right] \\
&\overset{(a)}{=} \mathbf{F}(\boldsymbol{\theta}),
\end{aligned} \tag{A.12}$$

where (a) follows from Lemma A.1. Thus $\mathbf{F}(\boldsymbol{\theta}) = \nabla^2_{\boldsymbol{\theta}'} D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\cdot) \| q_{\boldsymbol{\theta}'}(\cdot)\right)\big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}}$. □

Corollary A.2.. **[Fisher information and the 2nd order approximation of KL divergence]** *Given a distribution $q_{\boldsymbol{\theta}}(\cdot)$ paramertized by $\boldsymbol{\theta}$, consider perturbing the parameter by $\boldsymbol{\epsilon}$ such that $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\epsilon}$, the KL divergence[5] of the two distributions is:*

$$D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\cdot) \| q_{\boldsymbol{\theta}'}(\cdot)\right) = \frac{1}{2}\boldsymbol{\epsilon}^{\top}\mathbf{F}(\boldsymbol{\theta})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2). \tag{A.13}$$

5. It is easy to verify that the sign of $\boldsymbol{\epsilon}$ does not affect the result in Eq. A.13; in this regard, Fisher information may be considered as a *symmetric metric* in the distribution space.

Proof. Instantiate the likelihood function with the random variable $\mathbf{z}$. The second order Taylor expansion of $\log q_{\boldsymbol{\theta}'}(\mathbf{z})$ with regard to θ is given as:

$$\log q_{\boldsymbol{\theta}'}(\mathbf{z}) = \log q_{\boldsymbol{\theta}}(\mathbf{z}) + \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})^\top \boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^\top \nabla^2_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2). \tag{A.14}$$

We then expand the KL divergence between the two distributions by:

$$\begin{aligned}
D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\mathbf{z}) \| q_{\boldsymbol{\theta}'}(\mathbf{z})\right) &= \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\log q_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}'}(\mathbf{z})\right] \\
&\overset{(a)}{=} \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}[\log q_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z}) - \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})^\top \boldsymbol{\epsilon} - \frac{1}{2}\boldsymbol{\epsilon}^\top \nabla^2_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2)] \\
&= -\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})^\top \boldsymbol{\epsilon}\right] - \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\frac{1}{2}\boldsymbol{\epsilon}^\top \nabla^2_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})\boldsymbol{\epsilon}\right] + o(\|\boldsymbol{\epsilon}\|_2^2) \\
&= -\int \frac{\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\mathbf{z})^\top}{q_{\boldsymbol{\theta}}(\mathbf{z})} q_{\boldsymbol{\theta}}(\mathbf{z})\boldsymbol{\epsilon}\mathrm{d}\mathbf{z} - \frac{1}{2}\boldsymbol{\epsilon}^\top \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\nabla^2_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z})\right]\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2) \\
&\overset{(b)}{=} -\boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\theta}} \int q_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z} + \frac{1}{2}\boldsymbol{\epsilon}^\top \mathbf{F}(\boldsymbol{\theta})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2) \\
&= -\boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\theta}} \mathbf{1} + \frac{1}{2}\boldsymbol{\epsilon}^\top \mathbf{F}(\boldsymbol{\theta})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2) \\
&= \frac{1}{2}\boldsymbol{\epsilon}^\top \mathbf{F}(\boldsymbol{\theta})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2),
\end{aligned} \tag{A.15}$$

where (a) follows from Eq. A.14, and (b) follows from Lemma A.1 that $\mathbf{F}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[\mathbf{H}_{\log q_{\boldsymbol{\theta}}(\mathbf{z})}\right]$. Thus $D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}(\cdot) \| q_{\boldsymbol{\theta}'}(\mathbf{z})\right) = \frac{1}{2}\boldsymbol{\epsilon}^\top \mathbf{F}(\boldsymbol{\theta})\boldsymbol{\epsilon} + o(\|\boldsymbol{\epsilon}\|_2^2)$. □

This metric view offers a sensible interpretation on *common* trends of the Fisher information of $q_{\boldsymbol{\theta}}(\mathrm{y}|\mathbf{x})$ during network training. Relatively speaking:

- ***Low*** $\mathbf{F}(\boldsymbol{\theta})$: This implies that the gradient update will not change the prediction distribution much. It usually happens at (1) the *early phases* of training, where the prediction distribution is close to random, and a small variation to the parameter of the random will have little influence on the distribution; or (2) the *ending phases* (or converging phases), where the training is rather stabilized and the prediction distribution are close to true distribution, thus the gradients are close to zero, leading to a low Fisher information (also see Remark A.1).
- ***High*** $\mathbf{F}(\boldsymbol{\theta})$: This implies that even a small perturbation to the parameter can bring large discrepancy of the network prediction. Empirically, this typically happens when the network is (1) *learning new concepts* (*cf.* Achille et al. (2019)), or in other words *at the fitting phase* (in contrast to the generalization phase, *cf.* Shwartz-Ziv and Tishby (2017), especially on the relation to the gradient signal-to-noise ratio), or (2) *memorizing many hard or noisy examples* (*cf.* Jastrzebski et al. (2021)).

Often, $\mathbf{F}(\boldsymbol{\theta})$ will first increase, then drop, leading to a (skewed) bell-shaped trend during training. Thus the learning process appears to be crossing a barrier or a bottleneck.

The advantage of $\mathbf{F}(\boldsymbol{\theta})$ as a metric for the information content of neural networks is that it provides a dynamic (or variation) perspective compatible with the dynamic learning process, and it is relatively easy to compute. However, the limitation is also obvious — it can only serve as a *local* metric.

As we observe that the Fisher information of sparse scratch is much higher than sparse finetuning, a direct solution is to propose a data curriculum for sparse scratch, to ensure a more stable training near the initialization.

B. Experimental Details

B.1 Network Architectures and Training Specifications

Our code and re-implementation instructions is publicly available at this GitHub repository: https://github.com/ziyu-deep/Generalization-and-Memorization-in-Sparse-Training.

For Figure 1, 4, 5 and Table 2, the dataset used is CIFAR100 (Krizhevsky et al., 2009). The architecture used is ResNet (He et al., 2016) with depth of 32 and widen factor of 4. We initialize the network by Kaiming initialization (He et al., 2015). In network training, we use stochastic gradient descent (SGD) as the optimizer with a batch size of 128 with momentum of 0.9. The initial learning rate is set to be 0.1 and decays by 0.1 at the 100^{th}, 150^{th} and 200^{th} epochs. The number of training epochs for the dense network is 200, and for sparse finetune and sparse scratch is 250. The weight decay is set to be 0.0002. We use five different random seeds to do the training.

C. Supplement to Figure 3: Trends of Fisher Information

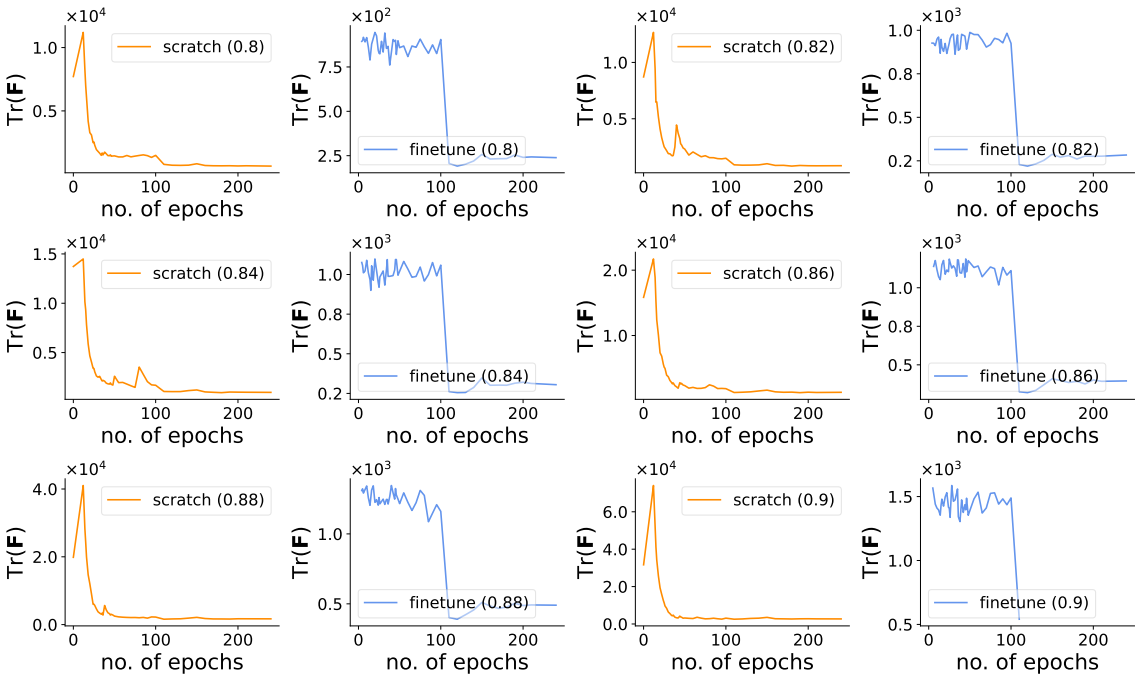


Figure 11: **Dynamics of Fisher Information** (Generalization Regime). The experiments are conducted on CIFAR100 with ResNet32 for sparse scratch and sparse finetuning with sparsity ratio varying from 0.80 to 0.90, corresponding the generalization regime in Figure 1.

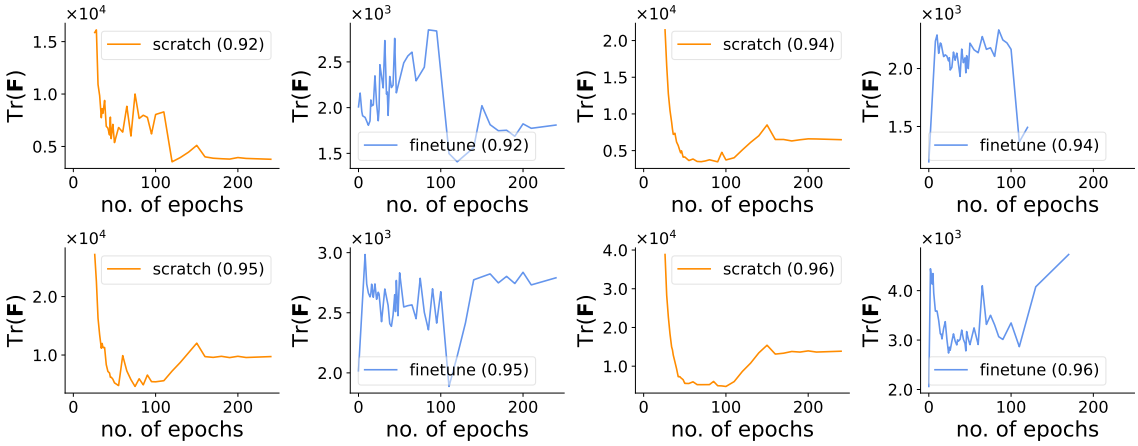


Figure 12: **Dynamics of Fisher Information** (Optimization Regime). The experiments are conducted on CIFAR100 with ResNet32 for sparse scratch and sparse finetuning with sparsity ratio varying from 0.92 to 0.96, corresponding the optimization regime in Figure 1.