# On the Relationship Between Model Training Dynamics and Early Pruning Periods

**Elvis Nunez**
Department of Electrical and Computer Engineering
University of California, Los Angeles
elvis.nunez@ucla.edu

**Stefano Soatto**
Department of Computer Science
University of California, Los Angeles
soatto@cs.ucla.edu

## Abstract

Contemporary deep learning models typically have billions of learnable parameters, requiring vast amounts of compute for training and inference. An established method for improving model efficiency is parameter pruning, whereby extraneous model parameters are discarded while trying to preserve model performance. Typically, such model compression is performed after the model has been trained. In this work, we aim to identify when a model becomes amenable to compression during training in order to realize the computational savings of model compression earlier. We showcase a phenomenon whereby an "early pruning period" occurs—a period during training where a model becomes amenable to pruning, prior to convergence. To help understand this behavior, we draw inspiration from recent work showing that model training undergoes two phases—the "memorization" and "forgetting" phases—and we show that the early pruning period often correlates with the transition between these two phases, suggesting that a large model capacity is only needed for the transient period of training, after which the model can be effectively compressed. We ground our study in discriminative computer vision applications and train multiple models across a spectrum of sizes, datasets, learning rate schedules, regularization strengths, and pruning criteria. We additionally propose a gradient-free metric that can be computed efficiently during training that also often correlates with the early pruning period. We show that we can identify a period early in the training of ResNet models trained on CIFAR where we can compress the model up to $90\%$ without incurring significant accuracy degradation.

## 1 Introduction

In this paper, we explore the question: "For how long during training does a model need to have a high capacity?" By "capacity," we mean the size of the model, i.e., its number of parameters, which we use as a proxy for the size of the hypothesis class that can be learned by the learning procedure. We are particularly interested in determining if there exists a point during training when a model's capacity can be reduced. We hypothesize that a model's large capacity may only be required to traverse the initial transient landscape of the loss surface, and may be reduced for the remainder of training, yielding a more efficient training procedure.

There are several means by which a model's capacity can be reduced, such as pruning [1–4], weight quantization [5–8] , and distillation [9, 10]. In this paper, we focus on parameter pruning due to its simplicity. Neural network pruning has been studied for several decades in classical machine learning, starting with [1] and [2] where parameters with small second-order derivatives were considered expendable and were subsequently pruned. Later, [3] re-introduced the idea of pruning to deep neural networks with a simple iterative magnitude-based criterion. These works highlighted the fact that neural networks can be pruned post-training—that is, once trained, a neural network can be made smaller without significantly compromising accuracy. Of note is [11], who showed that it is possible to identify a subset of a model's trained parameters such that, if trained in isolation from the outset, could have achieved the same performance as the full model. However, this requires first training a model to convergence before being able to identify the "winning ticket," i.e., the subset of parameters capable of being trained to the original model's performance.

Our experiments reveal an interesting phenomenon in which a model pruned early in training can match or exceed the performance of a model pruned later in training. We observe a correlation between this pruning period and the model's training dynamics, echoing several existing works that partition model training into distinct phases. As explored in [12], the generalization of a model decreases as its size increases, but after the model size passes a certain threshold, generalization increases once again. This "double-descent" phenomenon underscores the importance of a model's large capacity; whether this capacity is needed for the entirety of training is the focus of this paper.

In [13], it is suggested that networks trained with stochastic gradient descent (SGD) undergo two phases. In the first "empirical error minimization" phase, the network's layers increase the information on the labels, while in the second "compression" phase, the layers reduce the information on the input, i.e., they learn an efficient representation of the input. Similarly, [14] observe that the Fisher Information of a network's parameters undergo a "memorization" phase, in which the information that the weights contain about the data increases; afterwards, they undergo a "forgetting" phase, in which the Fisher Information decreases. These works allude to the idea of a phase shift during training driven by the model's capacity. Inspired by this, we aim to study early pruning periods through the lens of a model's training dynamics.

**Contributions:** We make the following contributions: 1) We showcase a phenomenon in which a model's capacity can be reduced early in training to greater effect than compressing the model after convergence. 2) We show that this early pruning period is often correlated with the model's Fisher Information, alluding to the idea of using this as a first-order metric for deciding when to prune a model. 3) We propose a gradient-free metric that can be efficiently computed during training in order to identify when a network has entered its early pruning period.

## 2 Related Work

**Model Pruning:** Compression of neural network models was first introduced in the 1980s [1], where a diagonal approximation to the Hessian matrix was used to assess parameter importance—parameters with a small value have little influence on the model's output and can hence be removed/pruned. Later, [2] considered a more robust computation of the inverse Hessian matrix to improve upon [1] which yielded a better selection of superfluous model parameters. A gradient-free pruning criteria was later considered in [3], where model parameters with small magnitudes (magnitude-based pruning) were iteratively pruned from a model. These methods performed unstructured model pruning, where pruned model parameters did not exhibit a coherent structure. On the other hand, structured pruning methods remove groups of parameters, often removing entire convolution filters, layers, or weight matrix rows/columns [4, 15–17].

**Early Pruning:** Our work is largely inspired by the Lottery Ticket Hypothesis [11], which found that models can be pruned early in training, but must first be trained to convergence before identifying which parameters to prune. Subsequent work [18] showed that models can be pruned once they become stable to SGD noise; however, detecting when this occurs remains a costly procedure. Many works have extended the Lottery Ticket Hypothesis, including work showing that winning tickets (the identified subset of weights that are kept after pruning) generalize to models trained on different datasets [19]. Several works have aimed to prune models prior to any training [20–22]; however, such methods have been shown to not perform as strongly as those pruned after some training [23]. Most similar to our work is [24], which introduces an iterative method to prune channels of convolutional neural networks early in training based on a "mask distance" metric coupled with a quantized training scheme. In our work, we prune models early using unstructured pruning in a one-shot (i.e., non-iterative) fashion, and reveal a connection between the early pruning period and the model's training dynamics.

## 3 Early Pruning Period

We consider a neural network $f(x;\theta)$ parameterized by $\theta \in \mathbb{R}^d$ that takes as input $x$. We train $f$ for a total of $T$ epochs and denote by $\theta_{0\rightarrow k}$, $k \in \{1, 2, \ldots, T\}$ the parameters after training for $k$ epochs when starting from random initialization $\theta_0$. In this paper, we consider the discriminative task of classification. We denote by $\text{Acc}\,(f(x;\theta_{0\rightarrow k}), D)$ the test accuracy of the model on held-out dataset $D$. We are interested in compressing $\theta_{0\rightarrow k}$ by pruning some of its weights, i.e., setting some of its values to 0. We denote by $\text{Prune}\,(\theta_{0\rightarrow k}, r)$ the pruning of $r\%$ of the parameters of $\theta_{0\rightarrow k}$.

Our goal is to identify a point in training, $t < T$, such that $\text{Acc}\,(f(x; \text{Prune}\,(\theta_{0\rightarrow t}, r)_{t\rightarrow T}), D) \simeq \text{Acc}\,(f(x;\theta_{0\rightarrow T}), D)$ where $\text{Prune}\,(\theta_{0\rightarrow t}, r)_{t\rightarrow T}$ denotes the training of parameters $\text{Prune}\,(\theta_{0\rightarrow t}, r)$ for an additional $T - t$ epochs. In words, we want to train a model for $t$ epochs, prune it, and then continue training it for the remaining $T - t$ epochs. Ultimately, we would like $t$ to be such that the performance of the pruned model, $\text{Prune}\,(\theta_t, r)_{t\rightarrow T}$, is similar to the performance of the non-pruned model, $\theta_T$. Below we describe our pruning methods and the metrics we use to help identify $t$. We emphasize that our pruning method does not entail fine-tuning the pruned model beyond the original training compute of $T$ epochs.

**Pruning:** We compress our model using unstructured pruning [3], which simply zeros out some of the elements in $\theta_{0\rightarrow k}$ by performing an element-wise multiplication with binary mask $\mathcal{M}_r$: $\text{Prune}\,(\theta_{0\rightarrow k}, r) = \theta_{0\rightarrow k} \odot \mathcal{M}_r$ where the percentage of zeros in $\mathcal{M}_r \in \mathbb{R}^d$ is $r$. $\mathcal{M}_r$ is constructed such that only the top $(100 - r)\%$ of weights of $\theta_{0\rightarrow k}$ are kept, and all other values are set to 0. In this paper, the "top $(100-r)\%$" of weights refers to the weights with the largest magnitude; however, as discussed in the next section, we also consider another metric to decide which parameters to prune. We focus on global unstructured pruning, but include results for local pruning in the Appendix. In global pruning, we keep the top $(100 - r)\%$ of parameters across the entire model. In local pruning, we keep the top $(100 - r)\%$ of parameters within each layer.

**Compression Signal:** Our task at hand is to judiciously decide *when* to prune a model during training, and *which* parameters to prune. Toward this end, we begin by performing a perturbation analysis on the predicted posterior distribution $p(y|x;\theta)$ modeled by the network $f(x;\theta)$. Given a perturbation, $\bar{\theta} = \theta + \Delta\theta$, one way of measuring the discrepancy between posterior distributions parameterized by $\theta$ and $\bar{\theta}$ is via the KL divergence, whose second-order approximation is given by

$$\mathbb{E}_{x\sim\hat{Q}(x)}[KL(p(y|x,\bar{\theta})\|p(y|x,\theta))] = \Delta\theta^T F \Delta\theta + o(\Delta\theta^2) \tag{1}$$

where $\hat{Q}(x)$ denotes the $\underline{\text{training}}$ dataset and $F$ is the Fisher Information Matrix (FIM) given by

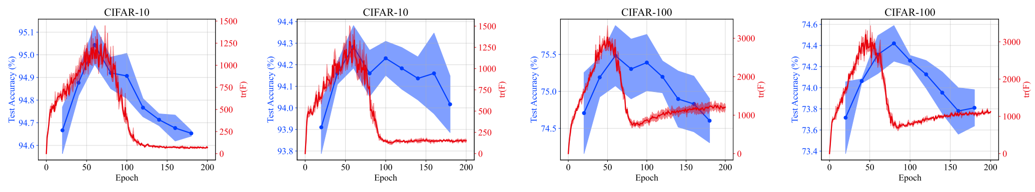
$$F = \mathbb{E}_{x\sim\hat{Q}(x)}\left[\mathbb{E}_{y\sim p(y|x;\theta)}\left[\nabla_\theta \log p(y|x;\theta) \nabla_\theta \log p(y|x;\theta)^T\right]\right]. \tag{2}$$

The FIM can therefore be interpreted as a local measure that quantifies how much the perturbation to a set of weights affects the predicted distribution. As such, while we focus on magnitude-based pruning due to its simplicity, we also consider pruning parameters with a low Fisher Information, which helps us decide *which* parameters to prune. However, computing the full FIM given by Eq. (2) is computationally expensive, so we compute only its trace, which is given by

$$\text{tr}(F) = \mathbb{E}_{x\sim\hat{Q}(x)}\left[\mathbb{E}_{y\sim p(y|x;\theta)}\left[||\nabla_\theta \log p(y|x;\theta)||^2\right]\right]. \tag{3}$$

The trace of the FIM, denoted by $\text{tr}(F)$, can be well-approximated using Monte Carlo estimation [14]. For a set of samples $\{x^{(i)}\}_{i=1}^N$ sampled from the training dataset, and corresponding $\underline{\text{predicted}}$ labels $\{\tilde{y}^{(i)}\}_{i=1}^N$, we use Monte Carlo sampling to estimate Eq. (3) by computing:

$$\text{tr}(F) \approx \frac{1}{N}\sum_{i=1}^{N} ||\nabla_\theta \log p(\tilde{y}^{(i)}|x^{(i)};\theta)||^2. \tag{4}$$



(a) (b) (c) (d)

Figure 1: **Early pruning periods correlate with tr$(F)$.** Blue points denote the test accuracies of models pruned at a $90\%$ sparsity level at the corresponding epoch $t$ and fine-tuned for $200 - t$ epochs. The red curve denotes the trace of the Fisher Information Matrix of the baseline (unpruned) models computed over the training set. The mean and standard deviation of three random seeds are plotted. (a): ResNet18 on CIFAR-10. Baseline accuracy: $95.3 \pm 0.06$. (b): VGG11 on CIFAR-10. Baseline accuracy: $94.7 \pm 0.07$. (c) ResNet18 on CIFAR-100. Baseline: $77.3 \pm 0.1$. (d): VGG11 on CIFAR-10. Baseline accuracy: $76.03 \pm 0.11$.

To help us understand *when* a model is most amenable to pruning, we draw inspiration from [14], where it was shown that models undergo two phases during training: a "memorization" phase and a "forgetting" phase. These two phases can be observed by measuring tr$(F)$ during training. During the memorization phase, which occurs first, the model's weights undergo an increase in the Fisher Information; afterwards, the model undergoes a "forgetting" phase, in which the Fisher Information of the weights decreases. Since the Fisher Information measures how much the predicted distribution is affected by a perturbation to the weights, weights with a small Fisher Information do not have a significant influence on outputs. The shift from the memorization phase to the forgetting phase therefore marks the period when parameters begin to become redundant. Hence, we explore whether tr$(F)$, computed over the training set via Eq. (4), can be used as a metric for deciding when to prune a model.

**Efficient Compression Signal:** While the trace of the FIM can be approximated using Monte Carlo estimation as in Eq. (4), it requires additional gradient computations beyond those computed for training the model, which will slow down training. To remedy this, we propose an alternative gradient-free metric that can be computed efficiently during training. In particular, we propose tracking the expected KL divergence between the predicted distribution, $p(y|x;\theta)$, and the uniform categorical distribution over $C$ classes, $p_u(y|x) = \frac{1}{C}$ for $y \in [C]$:

$$KL_{Uniform} \triangleq \mathbb{E}_{x\sim\hat{Q}(x)}\left[KL(p_u(y|x)\|p(y|x;\theta))\right]. \tag{5}$$

At initialization, the network's predicted distribution is nearly uniform; as training progresses, the learned distribution diverges from the uniform distribution, and computing Eq. (5) during training quantifies how this divergence evolves. We found that this metric also correlates with a model's early pruning period.

**Learning Rate Scaling:** In the standard train-prune-fine-tune framework [3], the learning rate used to fine-tune the pruned model is smaller than the learning rate used to train the dense model—typically $\frac{1}{10}$th of the original learning rate. However, since we apply pruning earlier in training, we found that simply thresholding the learning rate defined by the learning rate scheduler at the pruning epoch worked well. More precisely, the maximum learning rate used to train the pruned model at epoch $t$ was $\max(\Lambda, LR(t))$ where $LR(t)$ is the learning rate schedule's value at epoch $t$ and $\Lambda$ is a lower bound on the learning rate.

## 4 Experiments

We conduct extensive experiments on ResNet [25] and VGG [26] models on the CIFAR-10 and CIFAR-100 datasets [27]. Training details are provided in Appendix A. We ablate over model size, compression levels, learning rate schedules, and regularization in Appendix B. Our experiments in this section focus on global pruning; experiments on local pruning are provided in Appendix C.

### 4.1 Early Pruning Periods

**Early pruning.** Discriminative models can be effectively pruned after an initial transient period, often less than halfway through training. We trained a ResNet18 and VGG11 model on both CIFAR-10 and CIFAR-100 for 200 epochs, saving a model checkpoint after each epoch. After training, we loaded the model at various saved checkpoints, $t$, pruned 90% of the model's parameters using unstructured magnitude-based pruning, and then resumed training for the remaining $200 - t$ epochs with the learning rate computed as explained in Sec. 3. Our results are provided in Fig. 1, where we load and fine-tune globally-pruned models at every 20th epoch. This procedure was repeated three times, and we show results for the mean $\pm$ the standard deviation across the three trials (illustrated with the low opacity bands). As illustrated by the blue curve, pruning the model early, i.e., around epoch 60-70, yields the highest-accuracy pruned model. The baseline (non-pruned) CIFAR-10 ResNet18 model achieves an average test accuracy of 95.3%, while the ResNet18 CIFAR-100 model achieve an average of 77.3%. Hence, we observe that the model can be pruned early while incurring a minor accuracy degradation of about 0.2% on CIFAR-10, and 1.8% on CIFAR-100—though this is a more challenging task with an aggressive pruning rate. A more moderate level of sparsity can better preserve accuracy, as shown in Fig. 5 in Appendix B.2.

These results suggest that a large model capacity is needed for the model to memorize sufficient information about the training data, i.e., cross loss landscape bottlenecks as hypothesized in [14], but once this threshold is crossed, model capacity becomes expendable, and shedding some of these weights can yield more efficient training without a significant reduction in accuracy.

**When to prune?** The trace of the FIM [28] correlates with the performance of a pruned-model, suggesting its possible use as a metric for deciding when to prune a model during training. The red curve in Fig. 1 depicts the trace of the FIM of the dense model computed on the training data after every training epoch. We observe the "memorization" and "forgetting" phases first discovered in [14]. In the memorization phase, we see an increase in the amount of information about the dataset that is stored in the model's weights; afterwards, in the forgetting phase, we see a decrease of information. Crucially, we observe that the early pruning period occurs around the peak of tr$(F)$, that is, at the boundary between the memorization and forgetting phases. This suggests that a model may be most amenable to pruning when it is near the memorization/forgetting phase shift.

### 4.2 Fisher-based Pruning Criterion

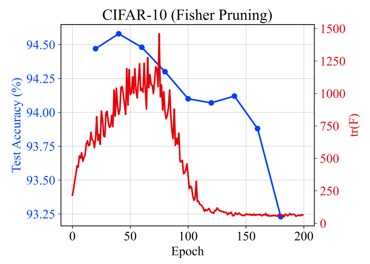
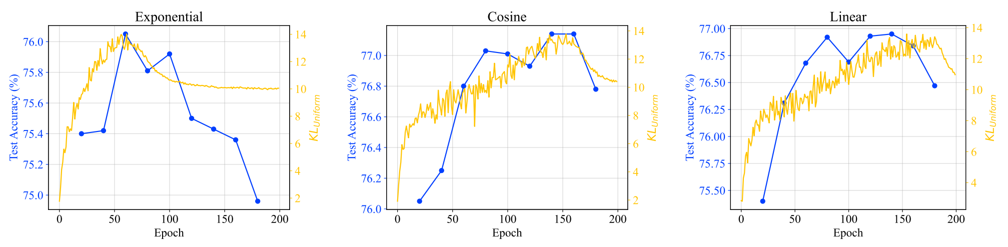


Figure 2: **Models pruned via a Fisher Information criterion exhibit an early pruning period.**

As discussed in Sec. 3, the Fisher Information of a set of the model's weights quantifies how much the predicted posterior distribution is affected by a perturbation to these weights. Hence, the Fisher Information of a set of weights can be interpreted as a measure of their synaptic strength, whereby weights with a low Fisher Information can be considered irrelevant. Up until now, we have used the magnitude of weights as our pruning criterion, removing weights with a small magnitude. Here, we experiment with pruning weights based on the their Fisher Information, removing weights with a small Fisher Information. In Fig. 2, we prune a ResNet-18 model at a sparsity level of 90% and prune it according to a Fisher Information criterion. We again observe an early pruning period that correlates with the FIM trace. This suggests that tr$(F)$ can be an effective pruning criterion. However, comparing the performance of this model to the model pruned with the magnitude-based criterion, i.e., Fig. 6a in Appendix B.3, we observe that the magnitude-based criterion achieves a higher test accuracy. While tr$(F)$ can often be a good proxy for assessing how much the model has memorized about its training data in aggregate, it may not be the best local measure for assessing parameter importance.

### 4.3 Efficient Compression Signal

To decide when to prune a model during training, we would like a metric that can be computed efficiently. As discussed above, tr$(F)$ can be an effective signal for deciding when to prune; however,



(a) (b) (c)

Figure 3: **The early pruning period correlates with the $KL_{Uniform}$ metric.** Models are trained on CIFAR-100 at a $90\%$ sparsity rate. The yellow curve denotes the $KL_{Uniform}$ metric of the baseline (unpruned) models computed over the training set. We observe that $KL_{Uniform}$ correlates with the test accuracies of the pruned models. (a): Models trained with an exponential learning rate schedule. Baseline accuracy: $77.4\%$. (b): Models trained with a cosine schedule. Baseline accuracy: $78.4\%$. (c): Models trained with a linear schedule. Baseline accuracy: $78.4\%$.

computing tr$(F)$ requires additional gradient computations beyond those computed to train the model, which will slow down training. To remedy this, we propose a more efficient metric, $KL_{Uniform}$ (see Eq. (5)), which forgoes gradient computations. This metric measures the divergence between the uniform distribution over the data's labels (the distribution that assigns equal probability to all labels) and the model's learned posterior distribution. In Fig. 3, we plot our $KL_{Uniform}$ metric of the unpruned model trained on CIFAR-100, along with the accuracies of the pruned models. Similar to the correlation observed with tr$(F)$ in Fig. 1, here we again see that the best performing pruned model tends to be the one pruned near the inflection point of the metric. While the inflection point of tr$(F)$ represents a shift from the memorization phase to the forgetting phase of training, the inflection point of $KL_{Uniform}$ marks a shift of the predicted posterior distribution toward the uniform distribution. This can also be interpreted as a "forgetting" phase, since a decrease in $KL_{Uniform}$ implies an attenuation of the model's overconfident predictions. Moreover, we observe that this metric also correlates with tr$(F)$, suggesting $KL_{Uniform}$ may be an efficient gradient-free proxy for tr$(F)$.

## 5 Conclusion and Future Work

In this paper, we aim to explore the question of *when* a model is most amenable to compression via pruning, particularly during training. We study model compression in the context of one-shot unstructured pruning, whereby we prune the model once *during* training, and fine-tune the model for the remaining epochs, staying within the original computational budget and yielding a pruned model without additional fine-tuning. We show a correlation between the trace of a model's Fisher Information, and when the pruned model performs best—what we refer to as "early pruning periods." By tracking a model's Fisher Information, tr$(F)$, during training, we found that a pruned model's performance is often best if it is pruned near the transition from the memorization phase to the forgetting phase. We posit that the evolution of the Fisher Information throughout training may serve as a low-fidelity signal that quantifies a model's learned redundancy. Our experiments highlight the need for a large model capacity at the outset of training, but after the initial transient period (i.e., near the memorization/forgetting phases), this capacity can be reduced in order to expedite training.

In addition to showing a correlation between the trace of the Fisher Information and the accuracy of pruned models at different stages of training, we also showed a correlation between our $KL_{Uniform}$ metric and when pruned models performed best. $KL_{Uniform}$ measures the KL divergence between the uniform posterior distribution assigning equal probability to data labels, $p_u(y|x) = \frac{1}{C}$, and the predicted posterior distribution defined by the model, $p(y|x, \theta)$. This gradient-free metric can be computed efficiently during training and may serve as a proxy for tr$(F)$, making it suitable for deciding when to prune a model.

Finally, we acknowledge that, due to computational constraints, the experiments presented in this paper were performed on small-scale models and datasets. Validating the claims presented in this paper on large-scale models and datasets remains an exciting future work with many practical applications that can make training more efficient and democratize access to large-scale training.

## References

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

[2] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[3] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[4] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[5] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[6] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[7] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. arxiv 2014. *arXiv preprint arXiv:1412.7024*, 2014.

[9] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL `https://api.semanticscholar.org/CorpusID:7200347`.

[10] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[12] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

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

[14] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856*, 2017.

[15] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[16] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019.

[17] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in neural information processing systems*, 36, 2024.

[18] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[19] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

[20] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[21] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[22] 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.

[23] Jonathan Frankle. *The Lottery Ticket Hypothesis: On Sparse, Trainable Neural Networks*. PhD thesis, Massachusetts Institute of Technology, 2023.

[24] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

[25] 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.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL `https://api.semanticscholar.org/CorpusID:18268744`.

[28] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.

## A Training Details

Our baseline models are trained on either CIFAR-10 or CIFAR-100 for 200 epochs. We use an SGD optimizer with 0.9 momentum and a maximum learning rate of 0.1 (without a warm-up). We use a batch size of 128. Unless stated otherwise, models are trained with a weight decay value of $5 \times 10^{-4}$ and with an exponential learning rate schedule that decays the learning rate by 0.97 after each epoch. When a model is pruned at epoch $t$, the pruned model is trained for an additional $200 - t$ epochs with the same learning rate schedule as its base model. However, as explained in Sec. 3, the pruned model is trained with a different maximum learning rate, namely $\max(\Delta, LR(t))$ where $LR(t)$ is the learning rate set by the learning rate schedule at epoch $t$, and $\Delta = 0.001$ is a lower bound hyperparameter. We choose this learning rate so that models pruned later in training are trained with a sufficiently large learning rate to recover performance, but not too large to compromise generalization; models pruned earlier will be trained with a learning rate similar to that which would be used by the base model.

## B Ablations

### B.1 Effect of Model Size

We observe that models across a spectrum of sizes (i.e., number of parameters) can be effectively pruned early in training. In Fig. 1, we showed results for ResNet18, which has approximately 11.2 million (M) parameters, and was originally intended for use on the ImageNet dataset and adapted for CIFAR. We further experimented with early pruning on ResNet20/32/44/56, each of which was designed specifically for the CIFAR datasets and range from having 0.27M parameters to 0.85M—all of which are smaller than the previous ResNet18 model. We provide the performance of these models in Fig. 4. For all model sizes, we again observe that the performance of pruned models correlates with tr$(F)$, where the early pruning period occurs near the memorization/forgetting phase shift.

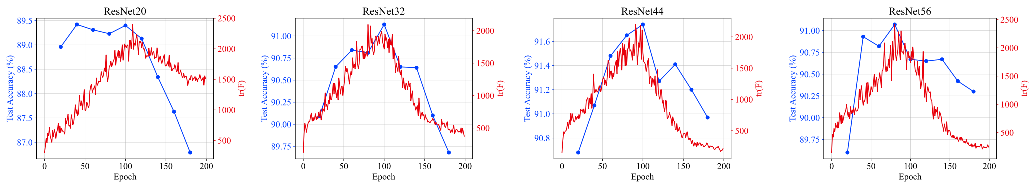


Figure 4: **Early pruning periods emerge across models of various sizes**. We train ResNet20/32/44/56 on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune 90% of the models' parameters and fine-tune for $200 - t$ epochs. We observe that there is a period early in training when pruning the model performs best. Baselines accuracies: ResNet20: $92.2\%$, ResNet32: $93.1\%$, ResNet44: $93.3\%$, 56: $92.5\%$.

### B.2 Varying Compression Levels

The benefits of pruning a model earlier in training are agnostic to the sparsity rate. Our experiments thus far have considered compressing a model by 90%—that is, removing 90% of its parameters with the smallest magnitudes. Next, we investigate how the performance of early-pruned models varies with sparsity levels. In Fig. 5, we provide results for a ResNet18 model pruned at multiple compression levels: 80%, 90%, and 95%. Models trained with a more moderate compression level exhibit the highest performance. Nevertheless, the model pruned early at a $95\%$ sparsity rate maintains a competitive performance compared to its baseline ($94.7\%$ vs. $95.3\%$). Additionally, we again observe that the early pruning period occurs near the memorization/forgetting phase shift.

### B.3 Effect of Learning Rate Schedules

The learning rate schedule can modulate the duration of the memorization and forgetting phases during training, as depicted by the red curves in Fig. 6, and hence can control when a model becomes amenable to pruning. Our previous experiments utilized an exponential learning rate decay, which

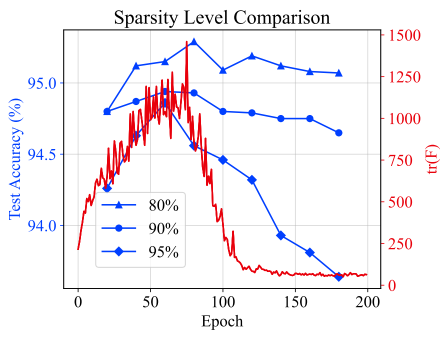
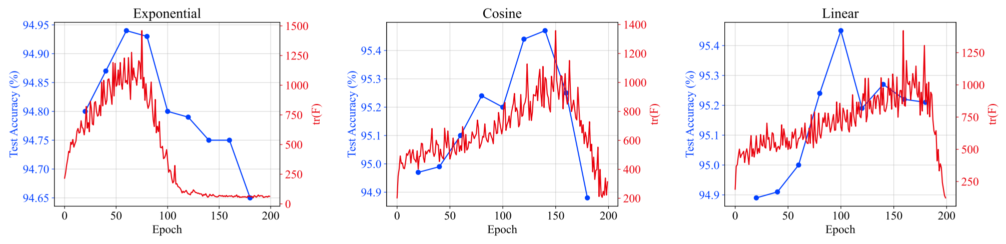


Figure 5: **Models pruned at various compression levels perform best when pruned near the peak of tr$(F)$.** We train a ResNet18 model on CIFAR-10 for 200 epochs. At every 20th epoch $t$, we prune either 80%, 90%, or 95% of the model's parameters via unstructured magnitude-based pruning. Performance of pruned models correlates with the peak of the dense model's FIM trace.



(a) (b) (c)

Figure 6: **The learning rate schedule modulates the length of the memorization phase and when the early pruning period occurs**. Models are pruned at $90\%$. (a): Models trained with an exponential learning rate schedule. Baseline accuracy: $95.3\%$. (b): Models trained with a cosine schedule. Baseline accuracy: $95.0\%$ (c): Models trained with a linear schedule. Baseline accuracy: $95.0\%$. When fine-tuning pruned models, we use the same schedule used to train the baseline models.

decayed the learning rate by 0.97 after every epoch. Here, we additionally train with cosine and linear decay schedules. In Fig. 6, we observe that the peak of the tr$(F)$ curve is dependent on the learning rate, with cosine and linear schedules reaching the forgetting phase later in training. This suggests that learning rate schedules determine the length of the memorization phase. In particular, as depicted in Fig. 7, the exponential schedule decays much faster than the cosine and linear schedules, suggesting that models enter the forgetting phase once the model has memorized sufficient information and the learning rate has become sufficiently small. Nevertheless, we continue to observe a correlation between the peak of tr$(F)$ and when the pruned model performs best.

### B.4 Weight Decay Regularization

The strength of correlation between a model's early pruning period and the trace of the FIM hinges on the model's ability to fit the data well. Our previous experiments employed L2 regularization ("weight decay") as a means of regularizing the network. The weight decay coefficient controls how fast model parameters are decayed (made smaller); larger coefficients impose stronger regularization, while smaller coefficients may lead to insufficient regularization. Our experiments have thus far used a coefficient of $5 \times 10^{-4}$, which we found to perform well. To study the effect of the regularization strength on the early pruning period, we retrained baseline models with coefficients of $5 \times 10^{-3}$ and $5 \times 10^{-5}$, and applied early pruning to these models. Our results are summarized in Fig. 8 (see Fig. 13 for a similar plot with local pruning). The baseline model trained with a coefficient of $5 \times 10^{-3}$ (Fig. 8a) underfit the data, while the the model trained with a coefficient of $5 \times 10^{-5}$ (Fig. 8c) overfit the data. All three cases exhibit an early pruning period; however, we observe a weak correlation between when the early pruning period occurs and the trace of the FIM. In the case

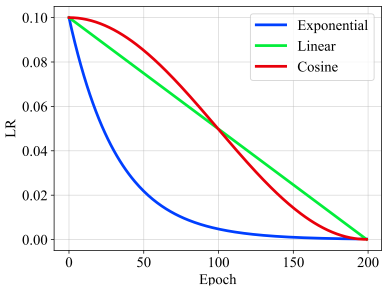
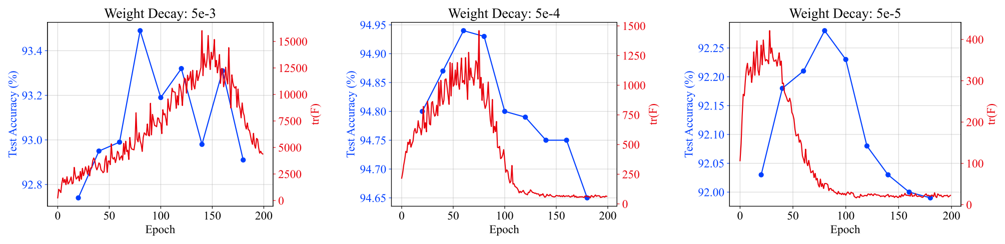


Figure 7: **We train models with exponential, cosine, and linear learning rate annealing schedules.** Pruned models follow the same learning rate schedule as their baseline model, with the max learning set as defined in Sec. 3.



(a) Underfitting

(b) Good fit

(c) Overfitting

Figure 8: **How well a model fits the data influences the correlation strength between a model's early pruning period and the trace of its FIM.** We trained a ResNet18 model on CIFAR-10 with varying weight decay coefficients. Each model was then pruned at varying points throughout training and fine-tuned for the remaining epochs. (a): Weight decay coefficient $5 \times 10^{-3}$; this baseline model underfit the data. Baseline: $93.4\%$. (b): $5 \times 10^{-4}$; this baseline model fit the data well. Baseline: $95.3\%$. (c): $5 \times 10^{-5}$; this baseline model overfit the data. Baseline: $92.8\%$.

of overfitting, we hypothesize that the weak regularization strength prompted the model to quickly memorize its training data and subsequently enter its forgetting phase early, leading to the discarding of parameters that may have become important later in training. Hence, the early pruning period occurs later—toward the end of the forgetting phase—once the appropriate superfluous weights can be identified. In the case of underfitting, the model receives larger gradient updates (as evidenced by the larger tr$(F)$ values, which are the norm of the gradients, as per Eq. (4)), and so tr$(F)$ may not begin to decrease until the learning rate is sufficiently small later in training. Hence, while the early weights to prune may have become evident earlier in training, tr$(F)$ was not able to capture when this occurred. Collectively, these results suggest that a model's early pruning period and its FIM trace are most strongly correlated when the model fits the data well.

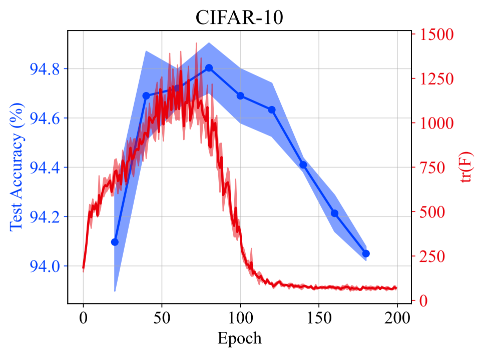
## C Local Unstructured Pruning

Here, we consider pruning parameters on a per-layer basis, keeping only top $(100-r)\%$ of parameters in each layer, where $r$ denotes the pruning rate. In Fig. 9, we recreate Fig. 1 using this local pruning scheme and again observe an early pruning period.

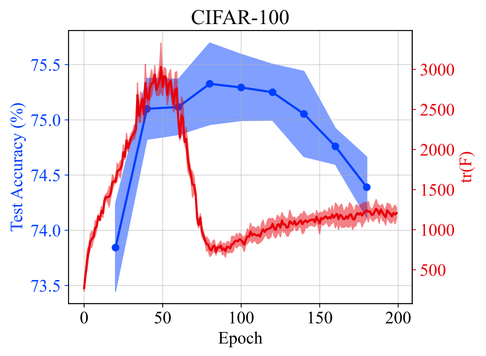
In Fig. 10, we observe that pruning models of various sizes via local pruning also produces early pruning periods that correlate with the trace of their FIM.

In Fig. 11, we show that models pruned via local pruning at various compression levels also exhibit an early pruning period that correlates with the trace of their FIM.

In Fig. 12, we show that models trained with different learning rate schedules that are pruned via local pruning also exhibit early pruning periods, and performance peaks near the transition between



(a)



(b)

Figure 9: **Early pruning periods correlate with tr$(F)$ when models are pruned at a local level**. Blue points denote the test accuracies of models pruned at a 90% sparsity level at the corresponding epoch $t$ and fine-tuned for $200 - t$ epochs. The red curve denotes the trace of the Fisher Information Matrix of the baseline (unpruned) models. The mean and standard deviation of three random seeds are plotted. (a): ResNet18 on CIFAR-10. Baseline accuracy: $95.3 \pm 0.06$. (b) ResNet18 on CIFAR-100. Baseline: $77.3 \pm 0.1$.

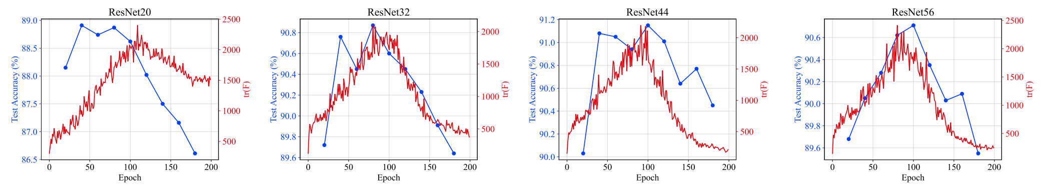


Figure 10: **Early pruning periods emerge across models of various sizes when pruned at a local level**. We train ResNet20/32/44/56 on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune 90% of the models' parameters and fine-tune for $200 - t$ epochs. We observe that there is an optimal time to apply pruning early in training. Baselines accuracies: ResNet20: 92.2%, ResNet32: 93.1%, ResNet44: 93.3%, 56: 92.5%.

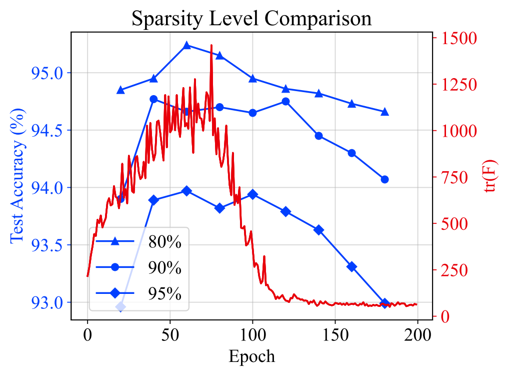
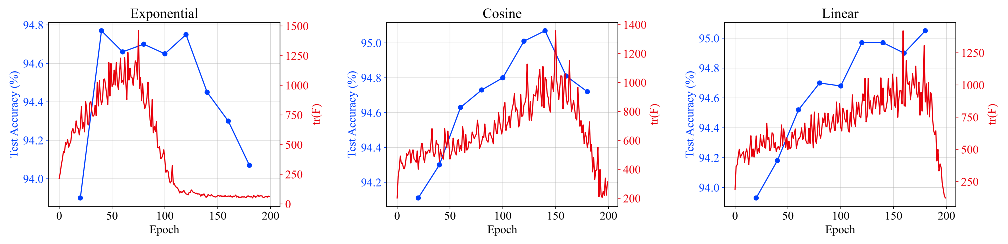
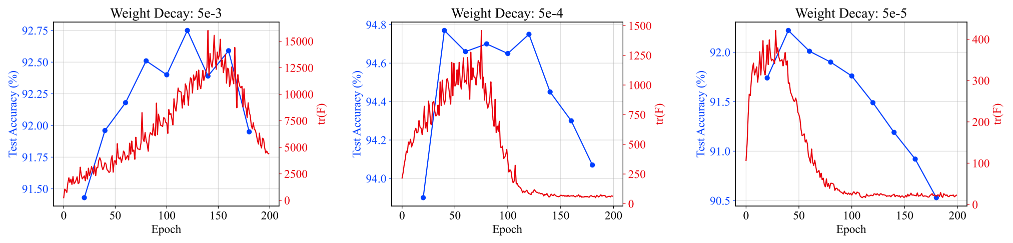


Figure 11: **Models pruned via local unstructured pruning at various compression levels perform best when pruned near the peak of tr$(F)$**. We train a ResNet18 model on CIFAR-10 for 200 epochs. At every 20th epoch $t$, we prune either 80%, 90%, or 95% of the model's parameters via unstructured magnitude-based pruning. Performance of pruned models correlates with the peak of the dense model's FIM trace.



(a) (b) (c)

Figure 12: **The learning rate schedule modulates the length of the memorization phase and when the early pruning period occurs for models pruned at a local level**. Models are pruned at 90%. (a): Models trained with an exponential learning rate schedule. Baseline accuracy: 95.3%. (b): Models trained with a cosine schedule. Baseline accuracy: 95.0% (c): Models trained with a linear schedule. Baseline accuracy: 95.0%. When fine-tuning pruned models, we use the same schedule used to train the baseline models.

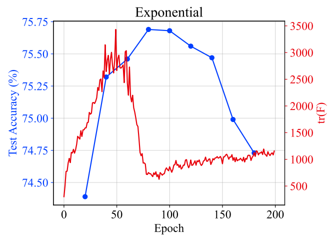


(a) Underfitting (b) Good fit (c) Overfitting

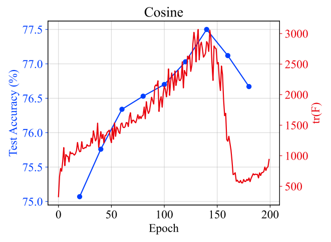
Figure 13: **The early pruning period and the trace of a model's FIM are correlated when the model is pruned at a local level**. We trained a ResNet18 model on CIFAR-10 with varying weight decay coefficients. Each model was then pruned at varying points throughout training and fine-tuned for the remaining epochs. (a): Weight decay coefficient $5 \times 10^{-3}$; this baseline model underfit the data. Baseline: 93.4%. (b): $5 \times 10^{-4}$; this baseline model fit the data well. Baseline: 95.3%. (c): $5 \times 10^{-5}$; this baseline model overfit the data. Baseline: 92.8%.

the memorization and forgetting phases. We obtain similar results for models trained on CIFAR-100, as shown in Fig. 14.

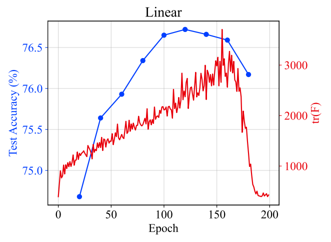
In Fig. 13, we train models with varying regularization strengths and then prune them via local pruning. In contrast to Fig. 8, here we see a stronger correlation between the pruned models' performance and their FIM trace. In Fig. 8, pruning is applied on a global scale, which means some layers may be pruned more aggressively than others. On the other hand, local pruning prunes layers uniformly. In the case of overfitting, the model's layers will specialize to fit certain attributes of the data, while the weaker regularization will lead to weights with an overall larger magnitude. We hypothesize that local pruning is more suitable in this regime since pruning on a global level may disproportionately prune layers with larger weight magnitudes. In the case of underfitting, the stronger regularization leads to weights with small magnitudes which may take longer to fit the training data, hence why we observe that the early pruning period occurs later in training. Again, pruning on a global level may lead to more agressive pruning of certain layers that have not yet fit the data well, so local pruning may be more appropriate in the underfitting regime as well.



(a)



(b)



(c)

Figure 14: **The learning rate schedule modulates the length of the memorization phase and when the early pruning period occurs for models pruned at a local level**. Models are pruned at $90\%$. (a): Models trained with an exponential learning rate schedule. Baseline accuracy: $77.44\%$. (b): Models trained with a cosine schedule. Baseline accuracy: $78.43\%$ (c): Models trained with a linear schedule. Baseline accuracy: $78.43\%$. When fine-tuning pruned models, we use the same schedule used to train the baseline models.