Early Model Training Dynamics as a Guide for Early Model Compression

Elvis Nunez
UCLA
elvis.nunez@ucla.edu

Abstract

Contemporary models in deep learning possess billions of learnable parameters requiring vasts amounts of compute for inference, and even more for training. An established method for improving model efficiency is neural network pruning, whereby extraneous model parameters are discarded while trying to preserve model performance. Typically, such model compression is performed iteratively after the model has been trained. In this work, we explore model compression during training in order to realize the computational savings of model compression earlier. Inspired by recent work showing that model training undergoes two phases—the "memorization" and "forgeting" phases—we show that a model is primed for pruning near the transition between these two phases, suggesting that large model capacity is only needed for the transient period of training. We ground our study in computer vision applications and train multiple models across a spectrum of sizes, datasets, learning rate schedules, regularization strengths, and pruning criteria. Our findings suggest that the optimal time to prune a model can occur early in training, often less than halfway through training. We introduce a gradient-free metric that can be computed efficiently during training and can be used as a signal for deciding when to prune a model. By compressing models earlier in training, the remainder of training can be made more efficient and yields a compressed model, circumventing the need for post-training compression and fine-tuning. We show that ResNet models trained on CIFAR-10 and CIFAR-100 can be compressed by 80% *one-third of the way through training without incurring significant accuracy degradation.*

1. Introduction

We are living in an exciting time when machine learning models can craft prose that echoes famous authors and can generate images that blur the line between artificial and human creativity. Driven by predictable scaling laws [12], the conventional approach for improving the performance of these models has been to feed them increasingly larger datasets while expanding their size and training for more iterations. Though the abilities of these models are remarkable, such scaling is becoming increasingly computationally demanding, restricting these methods of improvement to large corporations who have the financial capacity to undertake such endeavors. Moreover, the increasingly larger models—often on the order of having 10^9 to 10^{10} parameters—are restricted to running on high-capacity hardware accelerators such as GPUs and TPUs, making them impossible to run on commodity hardware.

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., the 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 identifying 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.

We are motivated by several existing works that investigate model capacity. Foremost, as explored in [24], the perplexingly small gap between training and test errors of large deep neural networks seems to defy conventional machine learning wisdom that larger models will memorize their training data and hence compromise generalization/test error. Despite this conventional wisdom, the current ethos is to train increasingly larger models, and as explored in [19], an interesting phenomenon occurs as model size continues to increase: generalization decreases as model size increases, but after the model size passes a certain threshold, generalization increases once again.

Several works have alluded to the idea of neural network training consisting of distinct phases. In [21], it is suggested the 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, [1] ob-

serve 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, and then undergoes 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. To this end, we explore whether a model's capacity can be reduced, via parameter pruning, once such a phase shift occurs.

Neural network pruning has been studied for several decades in classical machine learning, starting with [14] and [10] where pruning of parameters with small second-order derivatives were considered expendable. Later, [9] 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 [6], who showed that it was possible to identify a subset of a model's trained parameters such that, if trained in isolation from the outset, it 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.

A natural question extending [6] is whether winning tickets can be identified earlier in training, circumventing the need to train a model to convergence, and if so, how? In this paper, we make the following contributions:

1. We identify a point during training when a model's capacity could effectively be reduced without a significant reduction in accuracy.

2. We show the existence of a correlation between a model's Fisher Information and the accruacy of pruned models pruned at different stages in training.

3. We introduce a metric that can be efficiently computed during training in order to identify when a network becomes amenable to pruning.

4. We validate our method on several computer vision tasks across multiple models, datasets, pruning criteria, and regularization schemes.

2. Related Work

Model Pruning: Compression of neural network models was first introduced in the 1980s [14], 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, [10] considered a more robust computation of the inverse Hessian matrix to improve upon [14] which yielded a better selection of superfluous model parameters. A gradient-free pruning criteria was later considered in [9], 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 [5, 15, 17, 23].

Model Quantization: An alternative method to model compression is via model quantization, whereby model parameters are stored using lower bit-width precision, effectively reducing memory requirements and leveraging low-cost computations [3, 25, 26]. Recently, quantization has also been applied to large language models [4, 8, 16, 20] in an effort to reduce models' memory footprint and to make them more accessible across low-compute devices.

Early Pruning: Our work is largely inspired by the Lottery Ticket Hypothesis [6], which found that models can be pruned early in training—perhaps even before any training—but must first be trained to convergence before identifying which parameters to prune. Subsequent work [7] showed that models can be pruned once they become stable to SGD noise; however, detecting when this occurs remains a costly procedure. Extensive work has 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 [18]. Most similar to our work is [22], 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., not iterative) fashion, and reveal a connection between optimal pruning time and the model's transient period. Further, we introduce a simple metric that can be easily computed during training to decide when to prune the model.

3. Optimal Pruning Time

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 $\theta_{0\to k}$, $k \in \{1, 2, \ldots, T\}$ the parameters after training for k epochs when starting from random initialization θ_0. In this paper, we consider the discriminative task of classification. We denote by $\text{Acc}\left(f(x;\theta_{0\to k}), D\right)$ the test accuracy of main-dthe model on held-out dataset D. We are interested in compressing $\theta_{0\to k}$ by pruning some of its weights, i.e., setting some of its values to 0. We denote by $\text{Prune}\left(\theta_{0\to k}, r\right)$ the pruning [9] of $r\%$ of the parameters of $\theta_{0\to k}$.

Our goal is to identify a point in training, $t < T$,

such that $\text{Acc}\left(f(x; \text{Prune}\left(\theta_{0\to t}, r\right)_{t\to T}), D\right) \simeq \text{Acc}\left(f(x; \theta_{0\to T}), D\right)$ where $\text{Prune}\left(\theta_{0\to t}, r\right)_{t\to T}$ denotes the training of parameters $\text{Prune}\left(\theta_{0\to t}, r\right)$ 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, $\text{Prune}\left(\theta_t, r\right)_{t\to T}$, is similar to the performance of the non-pruned model, θ_T. Below we describe our pruning methods and the metric we use to identify t. We emphasize that our early pruning does not entail "fine-tuning" the pruned model beyond the original training compute of T epochs; rather, it trains a pruned model for the majority of the training budget, making training more efficient.

Pruning: We compress our model using unstructured pruning [9], which simply zeros out some of the elements in $\theta_{0\to k}$ by performing an element-wise multiplication with binary mask $\mathcal{M}_r$: $\text{Prune}\left(\theta_{0\to k}, r\right) = \theta_{0\to k} \odot \mathcal{M}_r$ 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 by magnitude of $\theta_{0\to k}$ are kept, and all other values are set to 0. We consider both global and local unstructured pruning. 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 θ 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 training dataset and F if 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, in addition to performing magnitude-based pruning, 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

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

As argued in [1], the trace of the FIM, denoted by $\text{tr}(F)$, can be well-approximated using Monte Carlo estimation for networks with residual connections [11]. In this paper, we consider architectures for classification tasks that utilize residual connections . For a set of samples $\{x^{(i)}\}_{i=1}^N$ sampled from the training dataset, and corresponding predicted labels $\{\tilde{y}^{(i)}\}_{i=1}^N$ we estimate Equation (3) as

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

To decide *when* to prune, we draw inspiration from [1], 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 the trace of the Fisher Information of the model's weights 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 use the trace of the Fisher Information, approximated by Eq. (4), as a metric for deciding when to prune a model. In particular, we compute Eq. (4) for every $\theta_{0\to k}$ for $k \in \{1, 2, \ldots, T\}$ and apply pruning at epoch t, when the trajectory of $\text{tr}(F)$ undergoes an inflection point, i.e., when the model transitions from the memorization phase to the forgetting phase.

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 may 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 distri-

bution diverges from the uniform distribution, and computing Eq. (5) during training quantifies how this divergence evolves. We found that this metric also correlates with when a model is most amenable to pruning.

Learning Rate Scaling: In the standard train-prune-fine-tune framework [9], 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. We adopt a similar scaling, however, since we apply pruning earlier in training, we use the following LR scaling:

$$LR = LR_{base}\left(1 - \frac{9t}{10T}\right) \tag{6}$$

where LR_{base} is the base learning rate used to train the dense model and t is the epoch at which we prune the model. Eq. (6) was constructed such that if we prune at epoch $t = T$, i.e., if we prune after the model has converged, then we scale the learning rate by $\frac{1}{10}$ as in [9].

4. Experiments

We study the early training dynamics of a neural network for discriminative classification tasks in the context of neural network pruning. Contrary to standard pruning techniques that first train a model for T epochs, and then iteratively prune it, our goal is to prune the model early in training and for the same total number of epochs, T. By conducting extensive experiments on ResNet [11] models trained on the CIFAR-10 and CIFAR-100 datasets [13], we aim to characterize and understand a model's training transient period—the initial phase of training when the model's weights undergo rapid change—and leverage this understanding to decide when to prune a model.

4.1. Unstructured Pruning of Discriminative Models

Early pruning. The optimal time to prune a discriminative model occurs early in training, often less than halfway through training. We trained a ResNet18 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, pruned 80% of the model's parameters using unstructured magnitude-based pruning, and then resumed training for the remaining epochs with the learning rate determined by Eq. (6) (e.g., when loading the checkpoint at epoch 60, we trained for an additional 140 epochs for a total of 200 epochs). Our results are provided in Figures 1a and 1b, 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., at epoch 60, yields the highest-accuracy pruned model. Compared to the non-pruned (dense) baseline models (the black curve), we observe that a fine-tuned model that is pruned early incurs only a minor accuracy degradation of about 0.5% on CIFAR-100, and a slight improvement on CIFAR-10, suggesting that early pruning can act as a regularizer—helping to mitigate overfitting. This suggests that the 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 [1], but once this threshold is crossed, model capacity becomes expendable, and shedding some of these weights can facilitate the learning process.

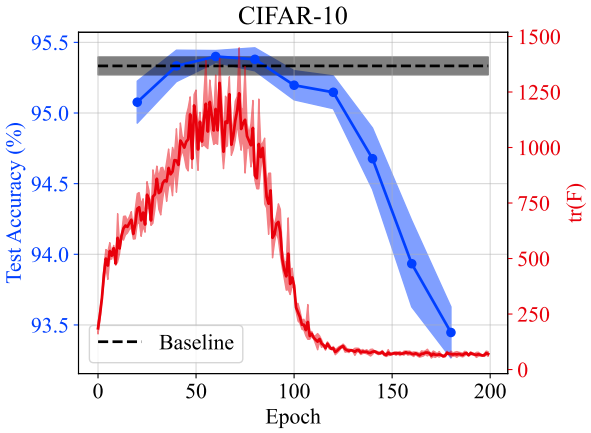
When to prune? The trace of the Fisher Information Matrix (FIM) [2] correlates with the performance of a pruned-model, and can be used as a metric for deciding when to prune a model during training. The red curve in Fig. 1 depicts the trace of the FIM of the dense model computed after every epoch of training. We observe the "memorization" and "forgetting" phases first discovered in [1]. 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 optimal pruning time occurs around the peak of $\text{tr}(F)$, that is, at the boundary between the memorization and forgetting phases. This suggests that a model is most amenable to pruning immediately after the transient period, as it shifts from memorizing its training data to forgetting it (see Fig. 9 for a similar figure with local pruning).

4.1.1 Effect of Model Size

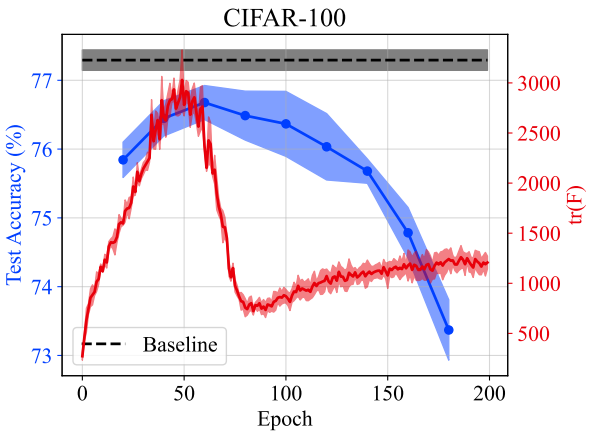
We observe that models across a spectrum of sizes (i.e., number of parameters) benefit from early pruning. In Fig. 1, we showed results for ResNet18, which has approximately 11.2 million (M) parameters, and was originally intended for use on the ImageNet dataset and adapted for CIFAR. We further experimented with early pruning on ResNet20/32/44/56/110, each of which was designed specifically for the CIFAR datasets and range from having 0.27M parameters to 1.7M—all of which are smaller than the previous ResNet18 model. We provide the performance of these models in Fig. 2. For all model sizes, we again observe that the performance of pruned models peaks when pruning is applied earlier in training.

4.1.2 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 80%—that is, removing 80% of its parameters with the smallest magnitudes. Next, we investigate how the performance of early-pruned



(a)



(b)

Figure 1. **Models pruned near the peak of tr(F) perform best.** (a): ResNet18 test accuracies on CIFAR-10. (b) ResNet18 test accuracies on CIFAR-100. Plotted lines depict the mean plus/minus the standard deviation for three random seeds. The black line denotes the baseline accuracies of the non-pruned models train for 200 epochs, while blue points denote the test accuracies of models pruned at the corresponding epoch t and fine-tuned for $200 - t$ epochs. The red curve denotes the trace of the Fisher Information Matrix (tr(F)) of the baseline models. High accuracies correlate with the peak of the FIM.

models varies with sparsity levels. In Fig. 3, we provide results for a ResNet18 model pruned at multiple compression levels: 70%, 80%, 90%, and 95%. As expected, models trained with a more moderate compression level exhibit the highest performance, while those compressed at a higher rate experience greater accuracy degradation. However, we again observe that the performance of pruned models is highest when the model is pruned close to the peak of the dense model's FIM trace.

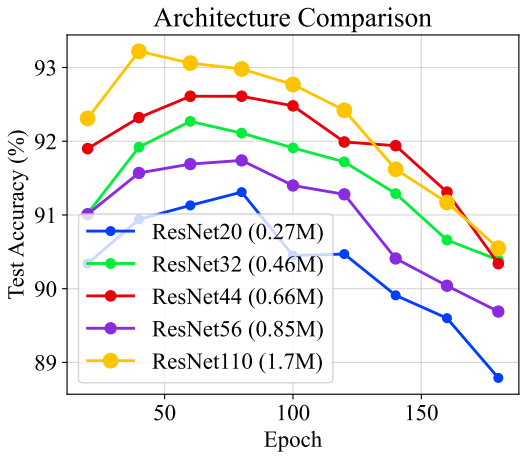


Figure 2. **Models of various sizes perform best when pruned early**. We train ResNet20/32/44/56/110 on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune 80% of the models' parameters and fine-tune for $200 - t$ epochs. We observe that the optimal time to apply pruning is early in training.

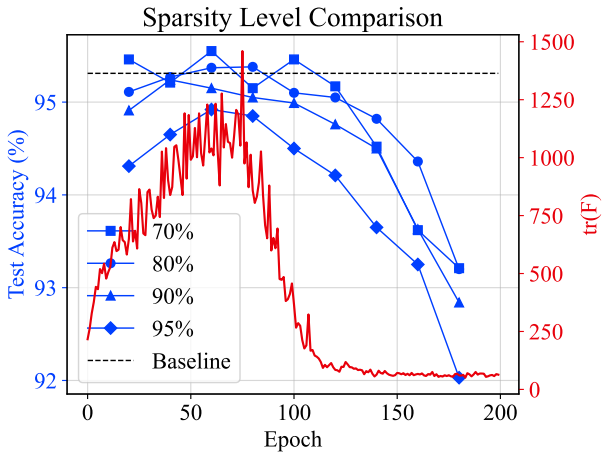


Figure 3. **Models pruned at various compression levels perform best when pruned near the peak of tr(F)**. We train a ResNet18 model on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune either 70%, 80%, 90%, or 95% of the model's parameters via unstructured magnitude-based pruning. Performance of pruned models correlates with the peak of the dense model's FIM trace.

4.1.3 Weight Decay Regularization

Early pruning hinges on a 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 im-

pose stronger regularization, while smaller coefficients may lead to insufficient regularization. Our experiments have thus far used a coefficient of 5×10^{-4}, which we found to perform well. To study the effect of regularization strength, we retrained baseline models with coefficients of 5×10^{-3} and 5×10^{-5}, and applied early pruning to these models. Our results are summarized in Fig. 4 (see Fig. 12 for a similar plot with local pruning). The baseline model trained with a coefficient of 5×10^{-3} (Fig. 4a) underfit the data and obtained the worst performance. From its FIM trace, we see that the model did not enter the forgetting phase until later in training; consequently, the learning rate used to fine-tune the pruned models at later epochs may have been too small for the model to learn effectively. On the other hand, the model trained with a coefficient of 5×10^{-5} (Fig. 4c) overfit the data, and the model entered its forgetting phase earlier in training. However, we observe that performance was not as high as the model trained with a coefficient of 5×10^{-4}, and subsequently its pruned models do not perform as well either. We believe that the model did not have sufficient time to memorize enough of the data, and suffered an accuracy loss as a consequence; nevertheless, we still observe that performance peaks around the transition from the memorization to the forgetting phase. In aggregate, these results suggest that a model needs to undergo a sufficiently long memorization period to learn effectively, and if pruning a model occurs later in training, then the learning rate must be scaled accordingly.

4.1.4 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 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. 5, 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.

4.1.5 Fisher-based Pruning

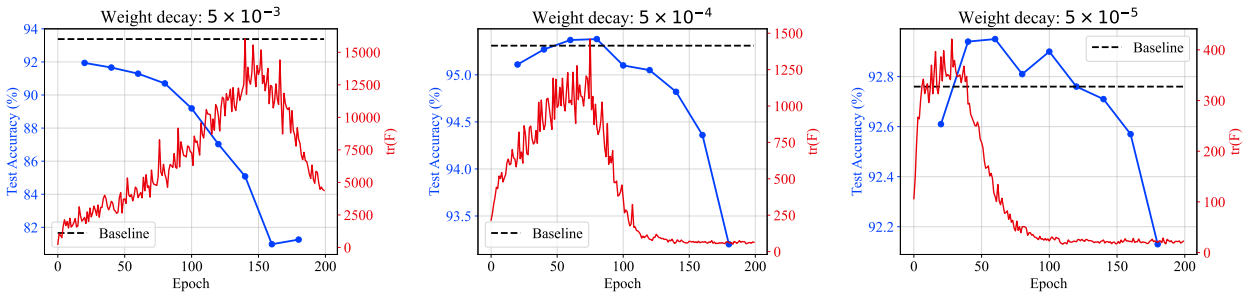
As discussed in Sec. 3, the Fisher Information of a set of the model's weights quantifies how much the predicted posterior distribution is affected by a perturbation to these weights. Hence, the Fisher Information of a set of weights can be interpreted as a measure of their synaptic strength, whereby weights with a low Fisher Information can be considered irrelevant. Previously, we 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. Our results for Fisher-based pruning are provided in Fig. 7, where we again observe that models pruned near the transition between the memorization and forgetting phases perform best. This suggests tr(F) can be an effective criterion for pruning when applied near the memorization/forgetting phase transition.

4.1.6 Efficient Compression Signal

Our pruning strategy occurs during training, and while training requires the computation of gradients of the loss computed over true labels, the gradients for tr(F) are computed for the loss over predicted labels (see Eq. (4)). This additional gradient computation can slow down training, so here we introduce a more efficient metric, $KL_{Uniform}$, as defined in Eq. (5). 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 learned posterior distribution defined by the model. In Fig. 8, we plot our $KL_{Uniform}$ metric on pruned models trained on CIFAR-100. Similar to the correlation observed with tr(F) in Fig. 6, here we again see that the best performing pruned model tends to be the one pruned at 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 find that this metric also correlates with tr(F), suggesting $KL_{Uniform}$ can 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. We train a baseline model for T epochs; then, we experiment with pruning the model at different epochs $t < T$ and train for an addi-



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

Figure 4. **How well a model fits the data largely influences how successfully we can prune the model.** 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 5×10^{-3}; this baseline model underfit the data. (b): 5×10^{-4}; this baseline model fit the data well. (c): 5×10^{-5}; this baseline model overfit the data.

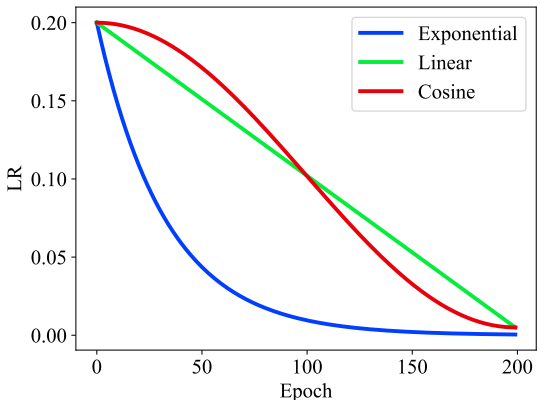


Figure 5. **We train models with multiple learning rate schedules**. We use exponential, cosine, and linear learning rate schedules to train our models.

tional $T - t$ epochs. This differs from contemporary pruning methods, whereby a model is trained to convergence, pruned, and then iteratively fine-tuned and pruned. In our setup, 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 when the peak of a model's Fisher Information trace occurs, and when the pruned model performs best. As previous work suggests, the Fisher Information reveals memorization and forgetting phases during training [1]. By tracking a model's Fisher Information during training, we found that a pruned model performs best if it is pruned near the transition from the memorization phase to the forgetting phase. Our experiments consistently showed a tendency for models to perform better when they were pruned near the memorization/forgetting phase transition; this correlation was observed across models of varying sizes, learning rate schedules, datasets, and pruning criteria. The phase shift between the memorization and forgetting phases marks the period when a model's training begins to discard superfluous information that it has memorized from the training data. Hence, it is reasonable to conclude that this is when a model is most amenable to pruning. However, we are cautious to conclude that the model's Fisher Information dictates this pruning behavior and we are reminded of the adage, "correlation does not imply causation." Rather, 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. By identifying when a model begins to shed this redundancy, we can simultaneously begin to compress the model.

In addition to showing a correlation between the trace of the Fisher Information and the accuracy of pruned models at different stages in 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 serves as a proxy for $\text{tr}(F)$ and can be used to decide when to prune a model.

Of interest for future work is applying our early pruning iteratively rather in a one-shot manner. Doing so would enable less aggressive pruning and may help mitigate accuracy degradation. Moreover, we have only explored one avenue of model compression: pruning. In future work, we can also consider quantizing a model early in training. Finally, the evolution of the Fisher Information of attention-based architectures is relatively unexplored, and in future work we aim to leverage this metric to compress such architectures.

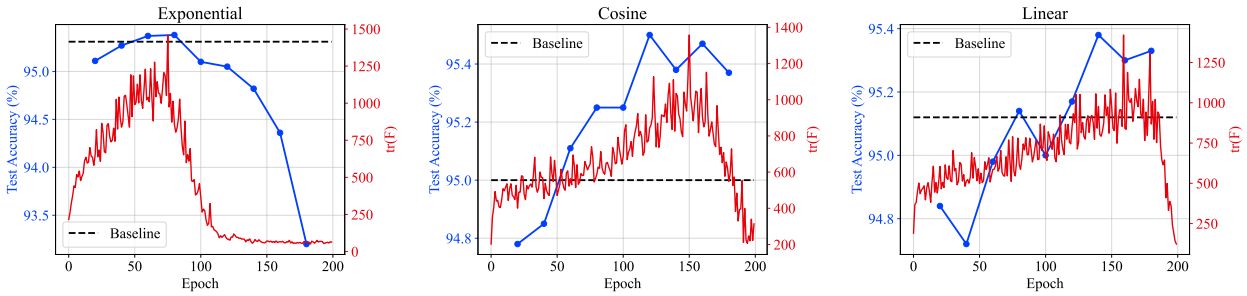


Figure 6. **The learning rate schedule modulates the length of the memorization phase and hence the optimal pruning time.** (a): Models trained with an exponential learning rate schedule. (b): Models trained with a cosine schedule. (c): Models trained with a linear schedule. When fine-tuning pruned models, we use the same schedule used to train the baseline models. Regardless of the learning rate schedule, models pruned near the end of the memorization phase tend to perform best.

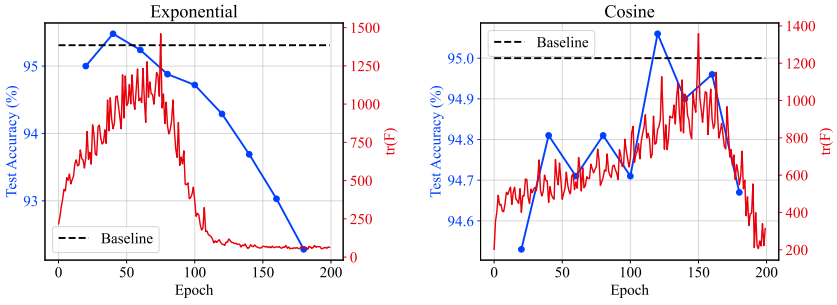


Figure 7. **Fisher-based pruning performs best when pruning is applied near the peak of tr(F).** (a): Model trained with an exponential learning rate schedule. (b): Model trained with a cosine learning rate schedule. Models are pruned according to a Fisher Information criterion, where we remove weights with a small Fisher Information.

References

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856*, 2017. 1, 3, 4, 7

[2] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000. 4

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015. 2

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[5] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in neural information processing systems*, 36, 2024. 2

[6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 2

[7] 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. 2

[8] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. 2

[9] 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. 2, 3, 4

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

[11] 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. 3, 4

[12] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for

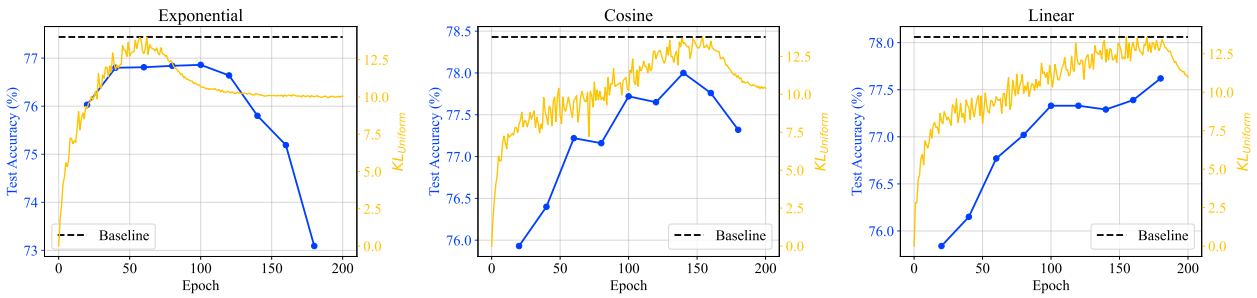


Figure 8. **The peak of $KL_{Uniform}$ correlates with the accuracy of pruned models.** (a): Models trained with an exponential learning rate schedule. (b): Models trained with a cosine learning rate. (c): Models trained with a linear schedule. Models are trained on CIFAR-100 and pruned at various epochs. The yellow curve denotes the $KL_{Uniform}$ metric and we observe that it correlates with the accuracies of the pruned models, i.e., the best performing pruned model is the one pruned around the peak of $KL_{Uniform}$.

neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

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

[15] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. 2

[16] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023. 2

[17] 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. 2

[18] 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. 2

[19] 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. 1

[20] Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022. 2

[21] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. 1

[22] 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. 2

[23] 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. 2

[24] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, feb 2021. 1

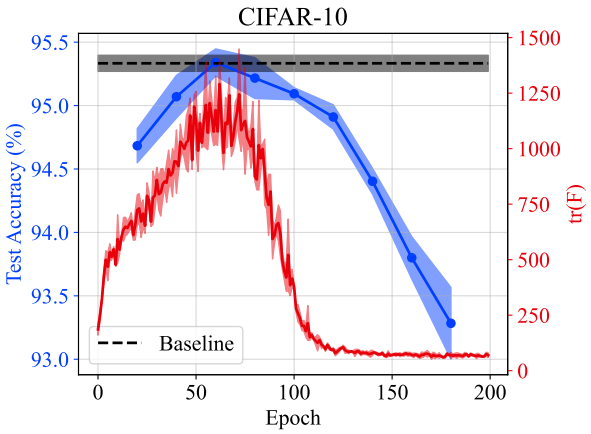
[25] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 2

[26] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2

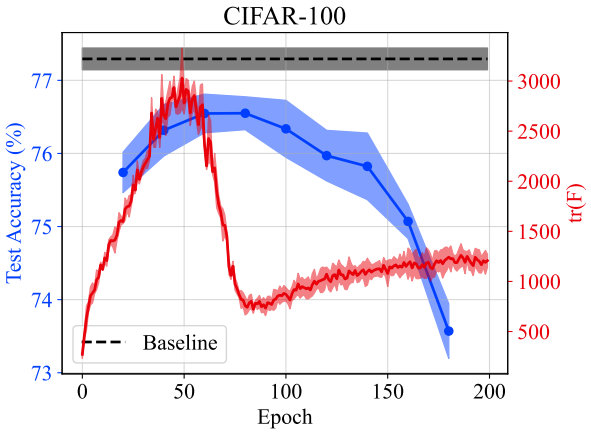
6. Appendix

6.1. Additional Results

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.



(a)



(b)

Figure 9. **Models pruned near the peak of tr(F) perform best when applying local pruning.** (a): ResNet18 test accuracies on CIFAR-10. (b) ResNet18 test accuracies on CIFAR-100. Plotted lines depict the mean $\pm$ the standard deviation for three random seeds. The black line denotes the baseline accuracies of the non-pruned models train for 200 epochs, while blue points denote the test accuracies of models pruned at the corresponding epoch t and fine-tuned for $200 - t$ epochs. The red curve denotes the trace of the Fisher Information Matrix (tr(F)) of the baseline models. High accuracies correlate with the peak of the FIM.

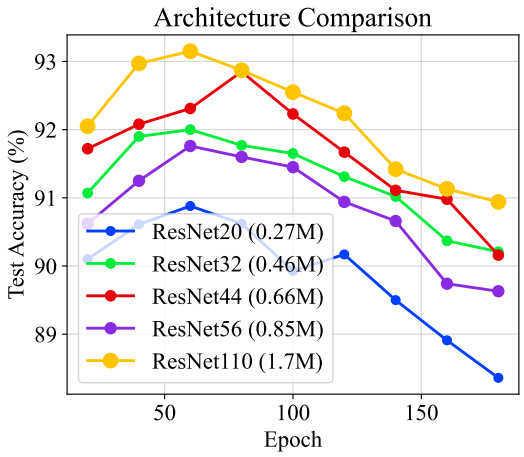


Figure 10. **Models of various sizes perform best when pruned early when applying local pruning**. We train ResNet20/32/44/56/110 on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune 80% of the models' parameters and fine-tune for $200 - t$ epochs. We observe that the optimal time to apply pruning is early in training.

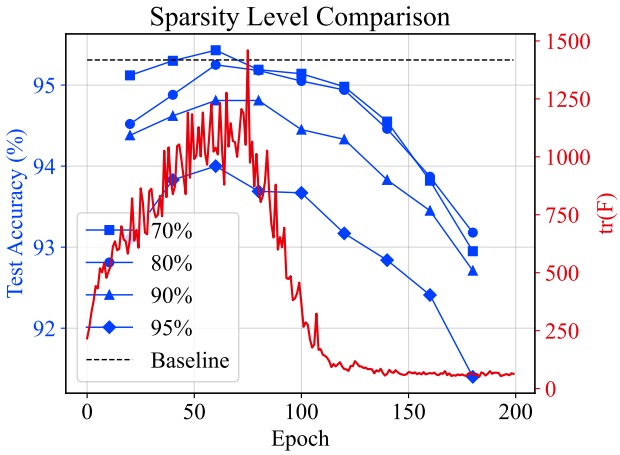
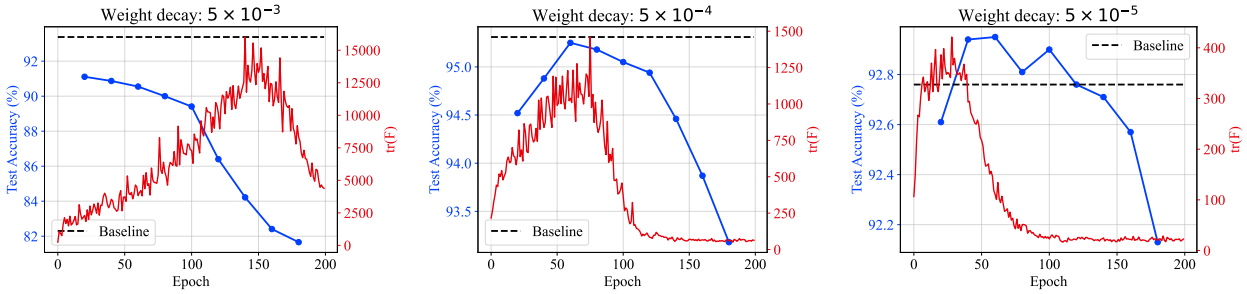
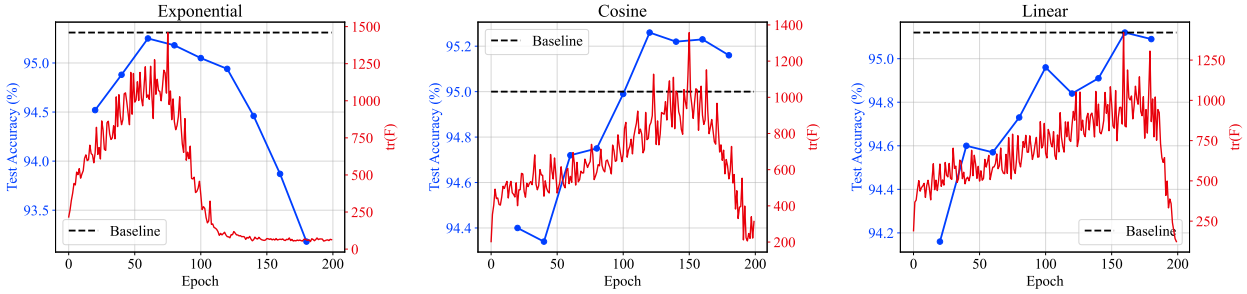


Figure 11. **Models pruned via local unstructured pruning at various compression levels perform best when pruned near the peak of tr(F)**. We train a ResNet18 model on CIFAR-10 for 200 epochs. At every 20th epoch t, we prune either 70%, 80%, 90%, or 95% of the model's parameters via unstructured magnitude-based pruning. Performance of pruned models correlates with the peak of the dense model's FIM trace.



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

Figure 12. **How well a model fits the data largely influences how successfully we can prune the model via unstructured local pruning.** 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 5×10^{-3}; this baseline model underfit the data. (b): 5×10^{-4}; this baseline model fit the data well. (c): 5×10^{-5}; this baseline model overfit the data.



(a) (b) (c)

Figure 13. **The learning rate schedule modulates the length of the memorization phase and hence the optimal (local) pruning time.** (a): Models trained with an exponential learning rate schedule. (b): Models trained with a cosine schedule. (c): Models trained with a linear schedule. When fine-tuning pruned models, we use the same schedule used to train the baseline models. Regardless of the learning rate schedule, models pruned near the end of the memorization phase tend to perform best.