UNCOVERING THE IMPACT OF LEARNING RATE FOR GLOBAL MAGNITUDE PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A common paradigm in model pruning is to train a model, prune, and then fine-tune or reinitialize and retrain in the lottery ticket framework. However, prior work has implicitly assumed that the best training configuration for model performance was also the best configuration for mask discovery. In this paper, we propose a simple idea: what if a training configuration which yields worse performance actually yields a better mask? To test this, we decouple the learning rates for mask discovery (LR_{find}) and mask evaluation (LR_{eval}). Using a ResNet-50 on Tiny ImageNet, we discovered the counterintuitive "decoupled LR phenomenon," in which smaller LR_{find} values lead to models which have lower performance, but generate masks with substantially higher eventual performance compared to using the same learning rate for both stages. We show that this phenomenon holds across a number of models, datasets, configurations, and also for one-shot structured pruning. Finally, we demonstrate that smaller LR_{find} values yield masks with materially different layerwise pruning ratios and that the decoupled LR phenomenon is causally mediated by these ratios. Our results demonstrate the practical utility of decoupling learning rates and provide clear insights into the mechanisms underlying this counterintuitive effect.

1 INTRODUCTION

There has been significant progress in deep learning in recent years, but many of the best performing networks are extremely large (Kolesnikov et al., 2019; Brown et al., 2020). The extremely large size of such models can be problematic due to the amount of compute and memory needed to train and deploy such models. One approach that has gained increasing popularity is model pruning: removing weights (unstructured pruning) or units (structured pruning) from a trained network in order to generate a smaller network with near-equivalent (and in some cases, better) performance (Blalock et al., 2020; Lin et al., 2020; Liu et al., 2017; He et al., 2019; Molchanov et al., 2019). These models can then be deployed on devices with far more constrained resources.

Another topic related to unstructured pruning that is of particular interest is the lottery ticket hypothesis (Frankle & Carbin, 2018). The lottery ticket hypothesis states that a large network contains a smaller subnetwork at initialization that can be trained to high performance in isolation, and provides a simple pruning method to find such winning lottery tickets (LTs). In addition to allowing the training of sparse models from scratch, the existense of LTs also suggests that over-parameterization is not necessarily required to train a network to high performance; rather, over-parameterization may simply be necessary to find a good starting point for training.

When pruning a network, one must decide how much to prune each layer. Should all layers be pruned equally? Or rather, should some layers be pruned more than others? Previous studies have shown that global pruning results in better compression and performance than layerwise (or uniform) pruning (Frankle & Carbin, 2018; Morcos et al., 2019). This is because global pruning ranks all weights/units together independent of layer, granting the network the flexibility to find the ideal layerwise pruning ratios (LPR), which we define as the percent that each layer is pruned.

One can frame most pruning methods as having two phases of training: one to find the mask, and one to evaluate the mask by training the pruned model (whether by rewinding and re-training or fine-tuning). Common practice has been to use the same hyperparameters for both finding and evaluating

Algorithm 1: Unstructured iterative magnitude pruning

Input: Model parameters $\boldsymbol{\theta}$, pruning percentage per iteration p, number of total pruning iterations N, mask-finding hyperparameters H_{find}, mask-evaluating hyperparameters H_{eval}
Output: Performance of trained pruned models

```
Initialize mask M_0 as all ones
Randomly initialize model weights θ_0
for i = 1...N do
    Train model θ ⊙ M_{i-1} to convergence using hyperparameters H_find to obtain θ_i
    Prune p% of parameters θ_i, the ones with the smallest magnitudes globally and have not been masked
      out by M_{i-1}, to create a binary mask M_i
    Rewind weights back to initial θ_0
end
Evaluate final mask or other masks of interest (M_i) by training θ ⊙ M_i with hyperparameters H_eval and
  record its performance
```

masks (Blalock et al., 2020). Here, we propose a new perspective for selecting hyperparameters for pruning based on a simple idea: use different hyperparameters for these two phases.

Common practice rests on the assumption that models with the best performance will generate the best masks, such that the optimal hyperparameters for mask generation and mask evaluation should be identical. But what if this assumption is flawed such that models which converge to *worse performance* yield *better masks*? In particular, since the best learning rate (LR) is often sensitive to many other characteristics of training, such as batch size or momentum, we focused on using different LRs for mask generation and mask evaluation. Our investigation led to the following contributions:

- Surprisingly, we found that the best LR to find the mask is often different from the best LR to train the regular model or to evaluate the mask (Figures 1 and 2). Counterintuitively, this means that a worse performance of the model at the time of pruning can generate a mask which leads to better performance for the final pruned model than a mask generated by a higher performance model.
- We show that this phenomenon is not an artifact of the particular setting we studied, and also occurs for structured pruning (Figure 4) and on other datasets as well as other variants of LTs, including late rewinding (Figure 3a), learning rate warmup (Figure 3b), and learning rate rewinding (Figure 3c).
- We found that different LRs_{find} led to materially different layerwise pruning ratios (LPR; Figures 5 and A4). Notably, we observed that higher LRs_{find} (which resulted in worse masks) consistently pruned early layers far more than those masks found with low LRs_{find} .
- Finally, we show that this phenomenon is causally mediated by changing LPR. When the LPR is fixed to a "good" LPR, higher LRs_{eval} again lead to better performance *and* better mask generation (Figure 6a). Alternatively, when the LPR is fixed to a "bad" LPR, small LRs_{find} now result in worse performance *and* worse mask generation (Figure 6b).

2 Methods

2.1 Modified pruning procedures

Our main experiments are based on the lottery ticket procedure from Frankle & Carbin (2018) along with follow up work (Renda et al., 2020) for unstructured pruning and Liu et al. (2017) for structured pruning. We use iterative magnitude pruning (IMP) for LT pruning and global pruning for both LT and structured because it has been shown to perform better than one shot and local pruning.

We define three different sets of hyperparameters: H_{standard}, which is used for standard training (i.e. training the model once, without pruning); H_{find}, which is used to find masks (i.e. the pre-training part of the pruning procedure); and H_{eval}, which is used for obtaining the final pruned model to be used for inference and evaluating how good a mask is. Note that our modified procedures can incorporate changes in any hyperparameters, but in this paper we focus on experiments with varying the learning rate, which we will denote LR_{find} and LR_{eval} .

Algorithm 2: Structured one-shot magnitude pruning

Input: Model parameters $\boldsymbol{\theta}$, pruning percentage per iteration p, mask-finding hyperparameters H_{find}, mask-evaluating hyperparameters H_{eval}
Output: Performance of trained pruned model

1 Randomly initialize model weights $\boldsymbol{\theta}$
2 Train model $\boldsymbol{\theta}$ using hyperparameters H_{find} to obtain $\boldsymbol{\theta}_{\text{final}}$
3 Prune $p\%$ of parameters $\boldsymbol{\theta}_{\text{final}}$ to create a binary mask M by pruning channels with smallest magnitude in the scale for the batch normalization layers, *i.e.*, γ
4 Evaluate the mask by training $\boldsymbol{\theta} \odot M$ with hyperparameters H_{eval} and record its performance

Unstructured pruning To account for these distinct sets of hyperparameters, we slightly modified the procedure for unstructured IMP as described in Algorithm 1. This procedure gives us the performance of the pruned model at the desired sparsity level with no additional compute required compared to the original LT procedure. To generate additional data points for figures in this paper, we separately evaluated some of the intermediate masks as well; this extra compute is used only for analyses and is not needed to obtain and use a pruned model in practice.

In this paper, we pruned weights with the smallest magnitudes throughout the whole network by $p = 20\%$ at each pruning iteration until we reached 99.9% pruned, for a total of 30 pruning iterations. For some side experiments, we also modified line 6 for techniques such as late rewinding and learning rate rewinding (Renda et al., 2020). The original LT algorithm is recovered when $H_{\text{eval}} = H_{\text{find}}$.

Structured pruning For structured pruning, we consider one-shot pruning using the scale of batch normalization layers as described in Algorithm 2. This also has no additional compute requirement compared to the original, and the original is recovered when $H_{\text{eval}} = H_{\text{find}}$.

2.2 Models and datasets

For our main results we used ResNet-50 (He et al., 2016) on Tiny ImageNet[1] (Le & Yang, 2015), and we also ran experiments with ResNet-18 on Tiny Imagenet and ResNet-50 on MiniPlaces (Zhou et al., 2017). We only pruned convolutional layers, so that the fully connected output layer of ResNets remain unpruned. Unless otherwise stated, we used SGD with 0.9 momentum and a batch size of 512. We trained for 110 epochs, decaying the learning rate with a gamma of 0.2 at epochs 70, 85, and 100. See Section A.1 in the appendix for full training details.

2.3 Random baselines

In pruning experiments, we compared the selected mask to two random baselines. The first, which we denote as "global random", selects a mask at random, uniformly throughout the whole network. This means that all layers have the same expected pruning ratio. The second, which we call "preserve LPR", also selects a mask at random, but with the same LPR as the winning ticket that we compare to; another way to think about this is that the mask of the winning ticket is randomly shuffled within each layer. Both types of random masks are evaluated with new randomly initialized weights.

3 Results

3.1 The decoupled LR phenomenon

As described in the introduction, previous approaches to pruning and lottery tickets have assumed the same learning rate for both finding and evaluating masks. But does the best LR for mask training necessarily make the best choice for mask generation? If so, we would expect to see that the best $\text{LR}_{\text{standard}}$, best LR_{find}, and best LR_{eval} would all be the same, which is what current practice assumes. However, if the best LR_{find} is different, that would mean existing pruning methods have found suboptimal masks; if LR_{eval} is different, then the final pruned models have been suboptimal.

[1] Use of these datasets is only for noncommercial, research purposes, and not for training networks deployed in production or for other commercial uses

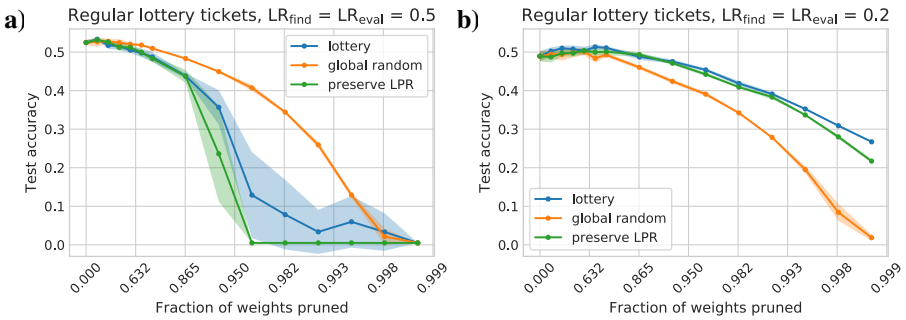

Figure 1: Test accuracy at various sparsity levels, with the regular lottery ticket procedure, i.e. LR_{find} = LR_{eval} . "Lottery" is the selected mask (winning ticket), "global random" is a mask selected at random globally, and "preserve LPR" is a random mask that has the same LPR as the winning ticket. (**a**) LR_{find} = 0.5 performed very poorly, even worse than the random baseline. (**b**) Curves we would expect from a properly functioning lottery ticket. The pruned performance is much better, even though training a regular model without pruning is better with LR as 0.5 than 0.2.

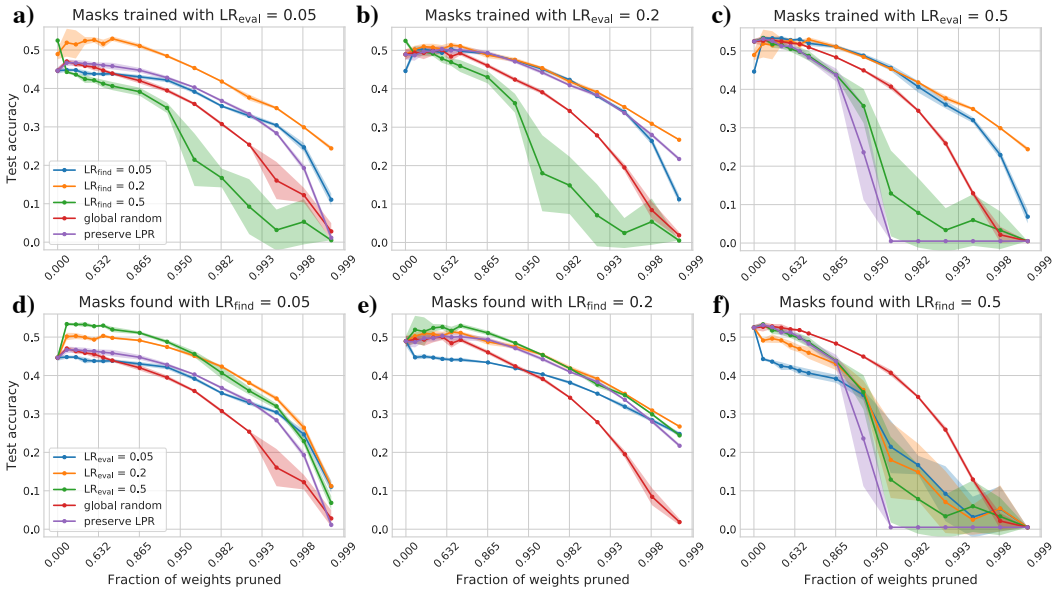

Figure 2: Test accuracy at different sparsity levels for different combinations of LR_{find} and LR_{eval} . Top row (a-c): given a constant LR_{eval} , we compare different values of LR_{find} . 0.5 is consistently and significantly worse than 0.2 and 0.05. Bottom row (d-f): given a constant LR_{find} , we compare different values of LR_{eval} . 0.5 is the best at lower sparsity levels, and 0.2 is the best at high sparsity levels. The points at 0% sparsity represent the unpruned model trained on the same LR_{find} as the rest of each line.

In standard training of the unpruned model, LR = 0.5 gives the best performance. Surprisingly however, we found that using LR = 0.5 in the original LT procedure (LR_{find} and LR_{eval} are both 0.5) results in the worst performance by far – even significantly worse than the random baseline! The failure of large LRs for lottery tickets has been pointed out previously (Liu et al., 2019), resulting in lottery tickets being trained with smaller LRs. This is illustrated in Figure 1a, where the winning tickets (blue) and the random baseline with the same LPR as the winning ticket (green) are both worse than the global random baseline (orange) by over 25% absolute accuracy at 96-98% sparsity. As a comparison, Figure 1b shows what a successful LT run typically looks like: the winning tickets should be significantly better than the global random baseline, and somewhat better than the random baseline that has the same LPR as the winning ticket.

This poor performance of the lottery tickets of LR = 0.5 is greatly improved by using a smaller LR of 0.2 (Figure 1b). However, 0.2 is a worse LR for training the standard unpruned model, as depicted by the points at 0% sparsity. If we had to use the same LR for everything, we would have to make a trade-off between a better unpruned model performance and better pruned model performance. Yet there is no reason to limit ourselves to the same LR for both stages, and we can instead use different LRs for different parts of the algorithm.

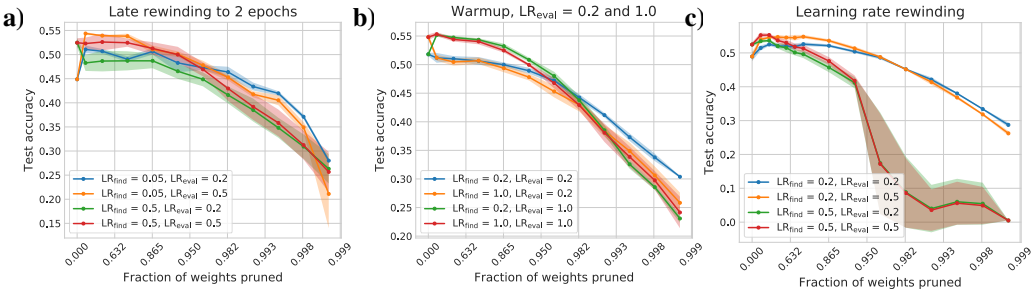

Figure 3: Test accuracy for various sparsity levels on variants of LT, with two values of LR_{find} and two values of LR_{eval} each. (**a**) Late rewinding to 2 epochs instead of initialization. (**b**) Learning rate warmup (**c**) Learning rate rewinding, where weights continue training from their values during pruning rather than rewinding back to an earlier value. In all three plots, the better LR_{find} for pruned models is the smaller one, despite performing worse on the unpruned model, and the best LR_{eval} starts high and then decreases.

In order to test whether it is better to decouple LR_{find} and LR_{eval} , we evaluated three different values each of LR_{find} , $LR_{eval} \in \{0.05, 0.2, 0.5\}$, for a total of nine combinations. The results are shown in Figure 2 in two different perspectives. In the top row, we compare different values of LR_{find} for a constant LR_{eval} , one plot for each of the values of LR_{eval} (a-c). These show that given a fixed LR_{eval} , a LR_{find} of 0.2 performs the best, closely followed by 0.05, and then 0.5 is by far the worst, despite performing the best on standard unpruned training. On the bottom, we compare different values of LR_{eval} for a constant LR_{find} , one plot for each value of LR_{find} (d-f). These show that given a fixed LR_{find} , it is better to use the largest LR_{eval} until the sparsity level reaches 98%, where the ideal LR_{eval} decreases to 0.2. In (f), the masks found by LR_{find} = 0.5 are so bad that no changes in LR_{eval} can save their performance.

These comparisons show that the most important change to improve from Figure 1a was to decrease LR_{find} . While 0.2 performs better than 0.05, they are close and both significantly better than 0.5. Meanwhile, LR_{eval} should be kept large, hence there is benefit to decoupling LR_{find} and LR_{eval} .

If we had followed common practice in the past, we would have swept over some LR values over standard training, found that 0.5 performed the best, trained LT with 0.5 as the LR_{find} , and gotten poor performance. This would have forced us to decrease the LR, allowing us to find reasonable masks, but at the expense of converged performance. However, by decoupling the LR_{find} and LR_{eval} , accuracy increased by over 30% at 99% sparsity over models with $LR_{find} = LR_{eval} = 0.5$, and the best performance in our nine combinations of LR_{find} and LR_{eval} outperformed any standard LT where the LR_{find} and LR_{eval} are the same by up to 3% (Figure 2).

Altogether, the results demonstrate two key findings: first, given a fixed LR_{eval} , it is better to use a smaller LR_{find} , and second, given a fixed LR_{find} , it is better to use a larger LR_{eval} , until you reach a certain sparsity level, where the ideal LR_{eval} drops slightly. Thus, it is important to decouple the LR_{find} and LR_{eval} in LT pruning.

3.2 The decoupled LR phenomenon is maintained across different models, datasets, configurations, and structured pruning

The strange failure case of LT at high LR_{find} raises a key question: how widely does this phenomenon generalize? Is this merely an artifact of the particular setting we studied (ResNet-50 on Tiny ImageNet)? To test this, we evaluated this phenomenon on different models and datasets as well as on several variants of lottery tickets.

Other models and datasets To expand beyond ResNet-50 on Tiny ImageNet, we also evaluated a ResNet-50 on MiniPlaces (Figure A1) and a ResNet-18 on Tiny ImageNet (Figure A2). The phenomenon is consistent in both: the best LR_{find} is smaller than the best $LR_{standard}$, and the best LR_{eval} starts high and decreases slightly at higher sparsity levels. Hence, ResNet-50 on Tiny ImageNet is not just a one-off case.

Other variants of lottery ticket For the same model and dataset as our initial results (ResNet-50 on Tiny ImageNet), we tried the following: using late rewinding, learning rate rewinding, and learning rate warmup. We ran late rewinding experiments from Renda et al. (2020), where instead

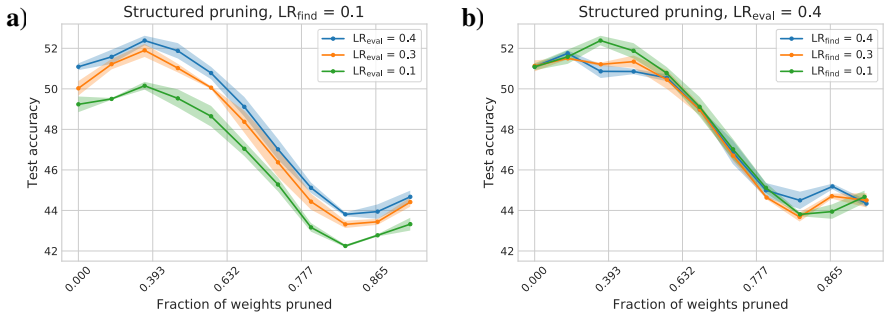

Figure 4: Small LRs_{find} also yield better masks in one-shot structured pruning. (**a**) Test accuracy for several LRs_{eval} for masks found with $\text{LR}_{\text{find}} = 0.1$. Consistently, training with a higher LR_{eval} leads to improved performance. (**b**) Test accuracy for several masks all trained with $\text{LR}_{\text{eval}} = 0.4$, and several LRs_{find}. Given a good LR_{eval}, small LRs_{find} consistently yield better masks.

of rewinding the weights to $\boldsymbol{\theta_0}$ in line 6 of Algorithm 1, we set the weights to their values at epoch 2. Here, we again see the same pattern, where the best LR_{find} is smaller than the best standard LR (Figure 3a). For learning rate warmup, we warmed up the learning rate linearly over 1000 iterations of linear increase from 0 to the specified LR_{find}. Here again, the effect of LR_{find} is the same, although it is more subtle (Figure 3b).

Our learning rate rewinding experiments also encapsulate the fine-tuning experiments (Renda et al., 2020). That is because both methods prune and then continue training the remaining weights from that point; the only difference is that learning rate rewinding trains at a large learning rate, and fine-tuning trains at a small learning rate (due to the decay in learning rate in the training phase before pruning). Thus, we extended learning rate rewinding to different LRs. We saw the same effect, with even better overall performance: taking a mask from a small LR_{find} and using learning rate rewinding to a high LR_{eval} improves accuracy by approximately 3 percentage points compared to weight rewinding (Figure 3c). This also explains why fine-tuning in Renda et al. (2020) did worse than learning rate rewinding: a larger LR_{eval} is better, and fine-tuning is effectively the same as learning rate rewinding but with a smaller and worse LR_{eval}.

Structured pruning We have demonstrated that our results in Section 3.1 are not merely an artifact of the precise setting we studied, but rather represent a more generic feature of lottery tickets across models, datasets, and configurations. However, this phenomenon may still be limited to only lottery tickets. To test this, we also evaluated whether this phenomenon is present for structured pruning on ResNet-18 Tiny ImageNet in which entire filters are removed. Once again, we found that the best combination is small LR_{find} and large LR_{eval}, where the best root performance is the large LR_{eval} (Figure 4). This beats the best constant LR by up to 2% in top-1 accuracy. These results demonstrate that the phenomenon of decoupling LRs does not just apply to lottery tickets, but rather applies to global magnitude pruning in general.

3.3 Layerwise pruning ratios causally mediate the decoupled LR phenomenon

In Figure 1a, we also observed that the "preserve LPR" random baseline was also quite bad. This led us to hypothesize that there may be some correlation between LPR and performance of lottery tickets. To assess this hypothesis, we looked at the LPR over different pruning iterations for various LRs. We see very different LPR for different LRs_{find} in Figure 5. Most strikingly, the large LR_{find} prunes the earlier layers much more. The imbalance starts early and does not correct itself as we prune more and more.

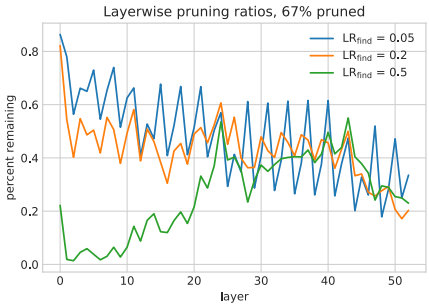

Figure 5: Layerwise pruning ratios at different LRs_{find}.

We have shown there is a clear difference in LPR across LRs_{find}, but is there a causal effect between LPR and performance of the pruned models? If this is the case, we should be able to change the performance of models just by constraining the model's LPR. To test this, we ran two experiments. Based on the results of Fig-

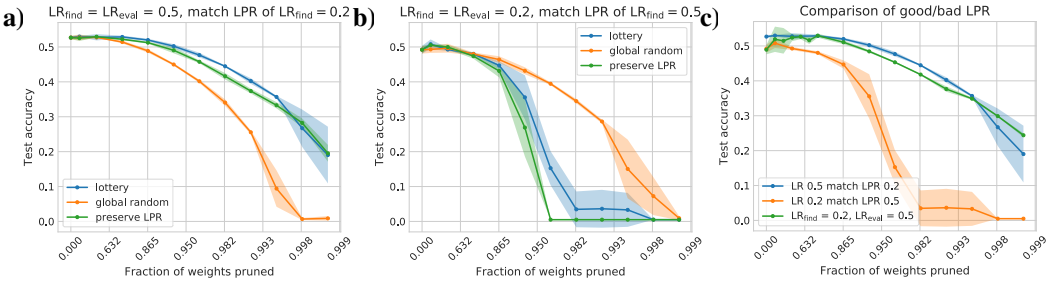

Figure 6: Test accuracy for various sparsity levels when pruned with some LR_{find}, but force layers to be pruned with LPR from a run with a different LR_{find}. (**a**) Running Algorithm 1 with $LR_{find} = 0.5$ but constraining LPR to be the same as the LPR from a run with $LR_{find} = 0.2$. Performance of the winning ticket improved drastically compared to Figure 1a. (**b**) Pruning with $LR_{find} = 0.2$ but LPR from a run with $LR_{find} = 0.5$ now performs very badly, worse than the random baseline. (**c**) Comparing the winning tickets of the two run configurations, along with a regular run of Algorithm 1 with $LR_{find} = 0.2$ and $LR_{eval} = 0.5$. The match-good-LPR run outperforms the analogous regular run of $LR_{find} = 0.2$, $LR_{eval} = 0.5$ as well.

ure 2, we consider the LPR of masks from $LR_{find} = 0.2$ as "good LPR", and the LPR of masks from $LR_{find} = 0.5$ as "bad LPR". We ran Algorithm 1 with the bad LR_{find} of 0.5 but forced the pruning step to prune a fixed amount each layer, specifically, according to the good LPR from the original 0.2 run ("match-good-LPR"). While the LPR are fixed, the masks within each layer were still determined by $LR_{find} = 0.5$. We ran the complementary procedure with a good LR_{find} of 0.2 but forced the LPR to be the bad LPR from the 0.5 run ("match-bad-LPR").

If LPR causally mediates the decoupled LR phenomenon, we would expect to see match-good-LPR do well and match-bad-LPR do poorly, despite being trained with LRs_{find} which typically result in the opposite. This is in fact what we observed: LR_{find} of 0.5 now clearly does better than the global random baseline (Figure 6a), and actually does better than smaller LRs_{find} as well (Figure 6c). Meanwhile, LR_{find} of 0.2 now does just as poorly as $LR_{find} = 0.5$ (Figure 6b). By manually changing LPR, we were able to account for the entire change in performance present in the decoupled LR phenomenon, thus showing the causal relationship between LPR and this effect.

To further confirm that fixing LPR nullifies the harms of a large LR_{find}, we experimented with fixing LPR to the same pruning ratio for all layers, equivalent to local pruning. The results (Figure A3) demonstrate the same pattern: once LPR is fixed, the larger the LR_{find}, the better the performance. Thus, the effect of LR_{find} can be summarized as: a small LR_{find} is needed to find good LPR, but a large LR_{find} is better to find the exact mask within each layer.

3.4 Early Pruning

As we show in Figure A4, for a high LR_{find}, LPR changes dramatically over training. On the other hand, with a low LR_{find}, the LPR is stable throughout training, with the exception of the first layer. Furthermore, the distance between the early mask's LPR and the actual mask's LPR decreases quickly and plateaus, likely due to the learning rate decay (Figure A5). This inspired us to experiment with pruning early, as in Yin et al. (2020). We found that early pruning does indeed work in some cases and can be even better than pruning at convergence. Pruning at either epoch 50 or 70 for $LR_{find} = 0.2$ actually improves performance compared to pruning at the end of training (Figure A6). There are two useful implications here. First, this is a sign that other hyperparameters besides learning rate follow the same phenomenon: the best settings to train the standard unpruned model (110 epochs) is not necessarily the same as the best settings to prune the model (50 epochs). Second, for the particular case of epochs spent training, this allows us to save compute. As mentioned in Section 2.1, pruning with decoupled H_{find} and H_{eval} requires no extra compute compared to the original LT procedure. In this case, H_{eval} decreases the compute used to find masks by 55% while also gaining a slight increase in test accuracy.

4 Additional Related Work

Our work is inspired by both the pruning literature (Li et al., 2020; Blalock et al., 2020) as well as by the lottery ticket framework Frankle & Carbin (2018). Magnitude pruning, in which weights or

filters are removed according to their magnitude, was first introduced in Han et al. (2015), and has had great success, prompting a large field of study, recently summarized by Blalock et al. (2020). Structured pruning approaches have had great success since such techniques can accelerate CNN during test-time without requiring specialiezd hardware and software support. Li et al. (2016) have demonstrated the potential of channel pruning for improving the efficiency of CNNs at test time. Later, many structural pruning methods were proposed to further improve the performance of structural pruning (Liu et al., 2017; He et al., 2019; Molchanov et al., 2019; Fan et al., 2020). Related to our work, Liu et al. (2019) have demonstrated the importance of layerwise pruning ratios over pre-trained weights for a wide range of structural pruning methods. Complementing their findings, our findings further suggest that while both weights and LPR are important for pruning techniques such as the lottery ticket framework, the value of weights is conditioned on good LPR, which is implicitly determined by hyperparameters when it comes to global magnitude pruning.

The lottery ticket hypothesis argues that over-parameterization is necessary to find a good initialization rather than for optimization itself (Frankle & Carbin, 2018; Frankle et al., 2019a). Since their introduction, a number of works have attempted to understand various aspects of this phenomenon, including their transferability (Morcos et al., 2019), the factors which make particular winning tickets good (Zhou et al., 2019), and linear mode connectivity (Frankle et al., 2019b). The lottery ticket hypothesis has also been challenged by (Liu et al., 2019), in part because of the requirement for small learning rates, as we address in this work.

Tanaka et al. (2020) observed that layer collapse, which they defined as entire layers being completely pruned, is a key issue that causes some pruned models to fail. This is along the same lines of the bad LPR that we observed, with a few key differences: they pruned to a higher sparsity level, we saw poor performance even before layer collapse, and skip connections in ResNets should make layer collapse less of an issue. Further, they claim that iterative pruning avoids layer collapse, but we see failure cases in iterative pruning.

In Li et al. (2020), the authors also decouple the learning rate between pre-training and fine-tuning, somewhat analogous to how we decouple the learning rate between finding and evaluating a mask; they are similar concepts applied to different tasks. As we analyze the impact of $\mathrm{LR}_{\mathrm{find}}$, they analyze the significance of the learning rate for fine-tuning on a different dataset.

5 Discussion

In this work, we proposed a simple idea: use a different learning rate for mask discovery and evaluation for pruning. By exploring this space, we uncovered the counterintuitive decoupled LR phenomenon (Section 3.1), in which using small learning rates that lead to *worse* performance yield *better* masks. We demonstrated that this effect is consistent across a variety of settings including going beyond lottery tickets, to structured pruning (Section 3.2). We then showed that small $\mathrm{LRs}_{\mathrm{find}}$ also lead to materially different layerwise pruning ratios in which early layers are pruned less than later layers (Figure 5). Finally, we showed that the decoupled LR phenomenon we identified is causally mediated by LPR (Section 3.3). While layerwise pruning ratios is not quite all you need, it does determine a very significant part of the final pruned performance and provides insights into the masks found by pruning. We hope that going forward, practitioners using pruned models will consider this new perspective and improve pruning methods with some simple additional sweeps of decoupled learning rates.

5.1 Limitations and future work

While we showed that the decoupled LR phenomenon is causally mediated by LPR, we did not find a clear answer to the question of why small $\mathrm{LR}_{\mathrm{find}}$ leads to better LPR. In addition, a method to derive a "good" LRP would be very useful. We hope to investigate these question further in future work.

Furthermore, it would be interesting to experiment with more datasets, architectures, and tasks beyond vision classification. Moreover, it would also be informative to evaluate other hyperparameters besides learning rate and total training epochs. We speculate that momentum and weight decay in particular might have an effect on LPR as well.

REFERENCES

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning?, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv*, pp. arXiv–2005, 2020.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. The lottery ticket hypothesis at scale. March 2019a. URL http://arxiv.org/abs/1903.01611.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis, 2019b.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2019.

Y. Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g8VkHFPH.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJem8lSFwB.

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*, pp. 2736–2744, 2017.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJlnB3C5Ym.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, 2019.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow, 2020.

Shihui Yin, Kyu-Hyoun Kim, Jinwook Oh, Naigang Wang, Mauricio Serrano, Jae-Sun Seo, and Jungwook Choi. The sooner the better: Investigating structure of early winning lottery tickets, 2020. URL https://openreview.net/forum?id=BJlNs0VYPB.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Conference on Neural Information Processing Systems*, 2019.