PRUNING CLOSE TO HOME: DISTANCE FROM INITIALIZATION IMPACTS LOTTERY TICKETS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Lottery Ticket Hypothesis (LTH) states that there exist sparse subnetworks (called 'winning' Lottery Tickets) within dense randomly initialized networks that, when trained under the same regime, achieve similar or better validation accuracy as the dense network. It has been shown that for larger networks and more complex datasets, these Lottery Tickets cannot be found in random initializations, but that they require lightly pretrained weights. In this paper, we aim to circumvent this pretraining phase by exploring techniques to improve the quality of the pruning masks that emerge during the *Mask Search* phase. We start by demonstrating that a simple choice of hyperparameters during the search can significantly impact the trainability of the sparse network, and that the trends that hold for searching are contradictory with the best hyperparameters for training. This leads us to explore learning distance as an explanation for this, which is influenced by hyperparameter selection, and demonstrate its impact via explicit regularization. We find that a lower learning distance in combination with limited sign flips during pruning results in sparse networks that are better trainable, and that after several pruning rounds there even emerge useful features as-is in the untrained sparse network. Finally, we combine these observations to significantly reduce the extraction cost of a highly sparse network.

1 INTRODUCTION

Whenever a neural network model is trained, the typical approach is to start with a dense overparameterized neural network. This technique has proven to lead to the best results as it allows the model to converge to a global optimum in the loss landscape when trained under Gradient Descent, rather than to a local optimum (Du et al., 2019). In contrast, starting training from a randomly sparse neural network typically fails to reach the same performance as an equivalent dense network (Li et al., 2016).

With the introduction of the Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2019), the notion that sparse networks cannot reach the same performance as a dense network when trained from scratch has been shattered. By using a costly procedure called Iterative Magnitude Pruning (IMP), it is possible to extract a sparse subnetwork from a randomly initialized dense network that generalizes equally as well when trained (termed a 'winning' ticket). However, this technique is only effective for small datasets and networks such as a LeNet-300-100 (LeCun et al., 2002) trained on MNIST, and starts to break down for more complex settings (Liu et al., 2019; Frankle et al., 2020a).

To remedy this, it was proposed to start from a slightly pretrained network, and extract a sparse network (termed a 'matching' ticket) from therein. Determining the required amount of pretraining to find such a ticket is often a difficult task requiring trial-and-error. One approach to do this was presented in Frankle et al. (2020a) which requires the final loss basin to be invariant to training noise, in order to find matching tickets. However, no theoretical explanation is given for this, nor is it shown whether this is a necessary or a sufficient condition.

Furthermore, as the IMP procedure involves repeatedly training networks to convergence, and then pruning with reduced pruning rates, it is computationally expensive to extract such a ticket. This led to several techniques being developed that limit the computational cost by reducing the number of involved training epochs (Zullich et al., 2021; You et al., 2020).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we aim to find 'winning' tickets irrespective of the complexity of the considered setting. To accomplish this, we first look at the impact different training hyperparameters have in the *Mask Search* phase, extending the batch size analysis of Maene et al. (2021). By reasoning about loss basins, we propose using Learning Distance (Nagarajan & Kolter, 2019) as a stand-in of stability to SGD noise, and show that limiting it can have a significant impact on the generalization, ultimately demonstrating significant gains in extraction time of the ticket.

Contributions:

- We discard the notion that pretraining is required to find winning lottery tickets in more complex settings, and instead explore various techniques in the *Mask Search* phase to improve the resulting ticket's trainability. (Section 4.1)
- By comparing various hyperparameter settings, we determine that there is a fundamental opposition between training a network for generalization, and training a network for magnitude pruning to achieve downstream sparse trainability. (Section 4.2)
- Building further upon the hyperparameter observations, we incorporate learning distance regularization in the *Mask Search* phase, and demonstrate its superiority to the default Weight Decay, and hyperparameter selection. (Section 4.2)
- Finally we show that in lottery tickets found with this strict regularization, purely by pruning by magnitude, useful features start to emerge in the randomly initialized weights, providing insight in why these tickets are more trainable. (Section 5.3)

2 Related Work

2.1 The Lottery Ticket Hypothesis

Analysis and extensions. Starting from when the Lottery Ticket Hypothesis (LTH), and the Iterative Magnitude Pruning (IMP) algorithm was introduced by Frankle & Carbin (2019), the phases that make up IMP have been analyzed and extended. Zhou et al. (2019) enumerates three distinct components in the *Mask Search* phase, namely the Mask Criterion, the Mask-1 action, and the Mask-0 action, and performs an ablation study across these components. Frankle et al. (2020a) discovered that lottery tickets were unable to be found in more complex settings, and remedied this by introducing late rewinding, which applies IMP on a lightly pretrained network (rewind point). Both Zullich et al. (2021) and You et al. (2020) introduced methodologies to speed up the *Mask Search* phase by limiting the training duration, respectively Accelerated Iterative Magnitude Pruning (AIMP) and Early-bird tickets. Paul et al. (2023) introduced a loss landscape geometry interpretation of IMP and used this to answer several questions related to the working of IMP.

Sanity checks for pruning masks. Several works question how much information a pruning mask carries beyond its layerwise sparsity. For methods that prune at initialization, Frankle et al. (2021) show that randomly shuffling the mask within each layer does not reduce accuracy, and Su et al. (2020) find that these methods can be replaced by random tickets with a suitable layerwise sparsity. Liu et al. (2022) further show that sparse networks with a purely random mask can match their dense counterpart when the network is sufficiently large. For IMP, Vischer et al. (2022) instead show that permuting the mask layerwise or model-wise destroys the trainability of the ticket.

Our analysis is also centered on the *Mask Search* phase, and more specifically on the hyperparameters which determine how far the network moves away from its initialization during this phase. This can be seen as a continuation of an experiment from Maene et al. (2021), which shows that for a sufficiently large batch size, winning tickets can still be found where previously late rewinding was required.

2.2 Features in Sparse Networks

Lottery ticket transfer. A number of studies have been conducted on which inductive features are contained in lottery tickets. Studies such as Morcos et al. (2019); Mehta (2019); Desai et al. (2021); Chen et al. (2021) train a ticket extracted for one dataset or data domain from scratch and test it on another target dataset or data domain. They all show that these trained lottery tickets can generalize as well as a trained dense model on the target dataset.

Untrained subnetworks. Zhou et al. (2019) first showed that selecting a mask can act as a form of training: applying such a supermask to a randomly initialized network achieves better than random accuracy without any weight updates. Ramanujan et al. (2020) scale this idea up by directly optimizing a score for every weight, and find untrained subnetworks in wide networks that match the accuracy of a trained dense network, introducing the Strong Lottery Ticket Hypothesis while Malach et al. (2020) prove that such strong lottery tickets exist in sufficiently overparameterized networks.

2.3 Distance from Initialization

Stability to SGD noise. Neural network solutions can be connected in the loss landscape via low-loss paths (Garipov et al., 2018; Draxler et al., 2018). Frankle et al. (2020a) relate the existence of winning tickets to a linear variant of this property: two copies of a network trained from the same (rewound) weights with different SGD noise should end in the same, linearly connected, loss basin. Measuring this requires training multiple copies of the network to convergence, which makes it an expensive criterion for choosing the amount of pretraining.

Learning distance. Defined as the ℓ_2 distance between the initialization and the final converged weights, this metric is typically used as a proxy for generalization. Li et al. (2020); Nagarajan & Kolter (2019); He et al. (2022) link a lower learning distance to limiting the model capacity and as such reducing overfitting on noisy labels. Conversely, Jiang et al. (2019) argue that there exists a negative correlation between generalization and learning distance. Explicitly penalizing the distance to a reference point is an established technique in transfer learning: L2-SP (Xuhong et al., 2018) regularizes a finetuned network towards its pretrained weights, using the same squared penalty as weight decay (Krogh & Hertz, 1991) but with a different anchor.

Sign flips. Zhou et al. (2019) found that the signs of the initial weights are a key component of a lottery ticket, more so than their magnitudes. Gadhikar & Burkholz (2024) attribute the advantage of Learning Rate Rewinding (Renda et al., 2020) over IMP to its ability to keep the sign flips that occur during training, whereas IMP reverts the weights, and thus their signs, to the initialization. This was then extended upon in (Gadhikar et al., 2025) by proposing a reparameterization that helps sparse networks trained from scratch to find good signs.

In this paper, we explore the learning distance in the context of the Lottery Ticket Hypothesis, both as a quantity, and an explicit optimization target via the ℓ_2 penalty during the Mask Search phase, with the rewind point as anchor.

2.4 Hyperparameters and Generalization

Batch size, momentum and learning rate. Smaller batches are known to generalize better, which Keskar et al. (2017) relate to converging to flatter minima, while Hoffer et al. (2017) attribute the generalization gap of large batches to the smaller number of weight updates. Momentum likewise improves generalization, particularly for larger batches (Jelassi & Li, 2022). Several works identify the ratio of learning rate to batch size as the scale of the SGD noise that governs these effects (Smith & Le, 2017; Jastrzebksi et al., 2017), which also motivates the linear scaling rule that keeps training dynamics equal across batch sizes (Goyal et al., 2017).

We find that this ratio also determines how far the network moves away from its initialization, and that the hyperparameters which improve dense generalization produce masks that are harder to train.

3 Background

3.1 The Lottery Ticket Hypothesis

Algorithm. Following Paul et al. (2022) we divide the training and extraction of a Lottery Ticket in three phases. **Phase I** is the *Pretraining* phase, in which a model is lightly pretrained to provide a more stable rewind point in order to limit pruning-induced accuracy degradation (Frankle et al., 2020a; Paul et al., 2023). The actual extraction of the sparse mask is in **Phase II**, the *Mask Search* phase. Here, a sparse network is iteratively trained, pruned via magnitude of the individual weights, and the unpruned weights are reverted to the rewind point. Finally once a sufficient sparsity has been

achieved, the Lottery Ticket is trained for a final time in the **Phase III**, the *Sparse Training* phase. (See Appendix, Section B for a full overview of the algorithm).

Phase impact on generalization. The final accuracy of a trained sparse network is influenced by each of these phases. **Phase I** sets the base weights of the model which will be pruned. By already pretraining, we avoid the noisy first phase of training (Frankle et al., 2020b), and induce some baseline generalization which can help the convergence during the second phase, and has been shown by (Frankle et al., 2020a) to be necessary for more complex settings. The actual connectivity of the network is determined by **Phase II** and arguably has the biggest impact. Vischer et al. (2022) illustrate this clearly by noticing that permuting the pruning mask either layer-wise or model-wise destroys the capacity for learning of the model. Finally, similar as they do for dense networks, training hyperparameters in **Phase III** can also affect generalization, and we show an example of this in the appendix, Section C.

Focusing on connectivity. Arguably, the largest body of research in neural network compression has been on pruning (Cheng et al., 2024). As such, our interest in achieving the best sparse network is not necessarily rooted in how to train a sparse network optimally, but rather in how connectivity can impact generalization. This means that we target **Phase II**, and how the choices we make there, can help finding better generalizing models. To isolate its impact, we need to limit the influence of **Phase I** and **Phase III**. Concretely this means that rather than using a pretrained model, our experiments all start from a randomly initialized model, and furthermore employ a fixed *Sparse Training* recipe throughout, irrespective of the changes made to **Phase II**.

3.2 Trainability and Generalization

Throughout this paper, we focus on the trainability of a lottery ticket, by which we mean its potential for generalization, expressed by validation accuracy. Concretely this means that we select a (set of) good-enough hyperparameter configuration(s) for the dense network, and then determine the accuracy degradation a lottery ticket would have when trained with the same hyperparameter selection. In effect, our goal is not to have a sparse network with the best possible absolute accuracy, but find techniques that limit accuracy degradation that then could be combined with state-of-the-art training recipes.

Hyperparameters and generalization We restrict our study to three hyperparameters with a well-documented impact on generalization. The *batch size* b controls the gradient noise from minibatch subsampling, where smaller batches are known to generalize better (Keskar et al., 2017). The *momentum* $\mu \in [0, 1]$ carries gradient information over from previous updates as an exponential moving average, and likewise improves generalization, particularly at larger batch sizes (Jelassi & Li, 2022). Finally, the *training budget* T (in epochs) bounds how much SGD noise accumulates during training. We consider $\mu \in \{0.0, 0.9\}$, $b \in \{100, 256\}$ and $T \in \{20, 200\}$ epochs, and show in the appendix (Section C that the observed trends hold for other values. All other hyperparameters are fixed: a learning rate of 0.2 with cosine annealing, a weight decay of 10^{-4}, and random cropping and horizontal flipping as augmentations.

3.3 Experimental Setup

Notation. We use θ to indicate the weights of a network, with an optional sparsity mask M. We use θ_0 to indicate the weights at initialization, while a superscript i indicates the pruning round in the *Mask Search* phase.

Models and datasets. In the main body of this paper we highlight the impact of our methodology with ResNet-34 trained on TinyImageNet. This baseline is chosen because it strikes the perfect balance between feasibility for in-depth experiments and complexity of the setting, given our limited computational resources. In the appendix we corroborate our main findings on several other settings, albeit less in-depth. Unless otherwise noted, we employ a pruning rate k of 20%, resulting in networks of sparsity of $\{67.23\%, 89.26\%, 96.48\%, 98.85\%\}$ after respectively $\{5, 10, 15, 20\}$ rounds of training, pruning, and rewinding. To avoid the effect of randomness, each experiment is the result of averaging over three random seeds.

Table 1: Dense network generalization under a variety of hyperparameter combinations.

	$\mu = 0.0$		$\mu = 0.9$	
	$b = 100$	$b = 256$	$b = 100$	$b = 256$
$T = 20$eps	54.06%	52.49%	57.09%	50.71%
$T = 200$eps	61.14%	59.33%	**63.74%**	61.57%

Table 2: Accuracy degradation of Lottery Tickets w.r.t. the dense baseline at various sparsities using different **Phase II** hyperparameters. Entries annotated with † comprised one or more runs that did not train beyond random chance (0.5%).

	$T = 20$eps				$T = 200$eps			
	$\mu = 0.0$		$\mu = 0.9$		$\mu = 0.0$		$\mu = 0.9$	
Sparsity	$b = 100$	$b = 256$	$b = 100$	$b = 256$	$b = 100$	$b = 256$	$b = 100$	$b = 256$
67.23%	-0.55%	**-0.19%**	-0.69%	-0.58%	-0.83%	-0.91%	-1.28%	-1.31%
89.26%	-1.76%	**-1.55%**	-2.75%	-2.10%	-2.93%	-2.18%	-27.79%†	-24.54%†
96.48%	-4.78%	-4.47%	-5.20%	**-4.41%**	-5.60%	-4.26%	-28.05%†	-25.90%†
98.85%	-9.42%	**-9.26%**	-9.98%	-9.33%	-11.00%	-8.47%	-48.65%†	-15.80%
Baseline	63.74%							

4 EXPERIMENTS

4.1 HYPERPARAMETER SELECTION IN PHASE II

Establishing the baseline. To extract our baseline Lottery Tickets, we chose to align the **Phase II** and **Phase III** training recipes, as would be the naive approach. Starting from the assumption that the best hyperparameters for *dense* training are also the best for *sparse* training, and that we always want to prune the best generalizing networks in **Phase II**, we conduct a small ablation test in Table 1 on the three axes $\{\mu, b, T\}$[1]. Here we find that $\{\mu = 0.9, b = 100, T = 200\text{eps}\}$ achieves the best performing *dense* network, meaning our baseline will use these hyperparameters for **Phase II + III**. As explained in Section 3.2, we aim to compare the trainability, rather than establishing the best possible generalization, so we fix these hyperparameters for *sparse* training throughout the rest of the paper.

Varying hyperparameters. In Table 2 we show the impact of different *Mask Search* recipes on the sparse network generalization. The most significant observation we make is that aligning the recipe of **Phase II** with the optimal Phase **III** recipe (i.e., using $\{\mu = 0.9, b = 100, T = 200\text{eps}\}$ in the *Mask Search* phase) resulted in the worst performance, with several of the runs converging to random chance (=0.5%) for the higher sparsity networks. Instead, for each of the hyperparameters, a value which increases dense network generalization, results in worse trainable sparse networks (compare Table 1 with Table 2). This implies that hyperparameters which aid in generalization during training, hinder the trainability for the downstream sparse network. Note that this observation is independent of the hyperparameters used in **Phase III** which we demonstrate in Section C. To find an explanation for this phenomenon, we turn towards the concept of SGD noise.

4.2 SGD NOISE AND LEARNING DISTANCE

The influence of noise. Frankle et al. (2020a) posits that in order for winning tickets to exist, the final loss basin – the attractor of the optimization process – should be invariant to the optimization noise. While adding more noise can help a model learn a more general solution, improving the validation accuracy, this also leads the model to travel further during optimization, leading to loss basins that are more difficult to find in a subsequent *Mask Search* iteration. As such, an obvious idea is to find loss basins that (1) are easier to converge to irrespective of sparsity, and (2) still generalize

[1] A larger ablation test is in Section C

Table 3: Learning Distance ($||\theta - \theta_0||_2$) under a variety of hyperparameter combinations.

	$\mu = 0.0$		$\mu = 0.9$	
	$b = 100$	$b = 256$	$b = 100$	$b = 256$
$T = 20$eps	53.45	31.62	127.40	105.10
$T = 200$eps	127.59	99.28	142.74	137.50

Table 4: Accuracy and Learning distance of a dense network when optimized with an additional target weighted by trade-off parameter λ.

	Accuracy				Learning distance			
Target	$\lambda = 0.0$	10^{-4}	10^{-3}	10^{-2}	0.0	10^{-4}	10^{-3}	10^{-2}
$\|\|\theta\|\|_2$	56.67%	63.74%	55.04%	0.50%†	575.19	142.74	113.34	113.05†
$\|\|\theta - \theta_0\|\|_2$		63.41%	62.91%	53.03%		115.03	35.82	10.69

well. We hypothesize that such loss basins do exist also close to initialization, but that these are skipped when purely optimizing for generalization.

Closeness to initialization. When we calculate the distance to the initialization for the dense networks trained under different hyperparameters (Table 3), we see that using 'suboptimal' hyperparameters during the training process actually restrain the search space during training, leading to solutions being found closer to the starting point. This can be more clearly seen in Figure 1 where we plot the evolution of learning distance throughout training. However, limited exploration can have the advantage that low-magnitude connections at the end of training, those deemed useless at convergence, are better aligned with those that are less important during the training process, leading to the sparse solutions being able to converge to the same loss basin as the dense solution.

Explicit regularization. To isolate the effect of this closeness to the initialization on lottery ticket generalization, we next integrate it in the *Mask Search* phase via explicit regularization. We explore two regularization targets: (a) $||\theta||_2$, the established weight decay technique, whose goal is to prevent overfitting by limiting weight growth and (b) $||\theta - \theta_0||_2$ which is simply the formulation of Learning Distance, and works similarly but uses the initialization as an anchor rather than the origin. These targets are added to the SGD loss during optimization with a trade-off factor λ. We demonstrate in Table 4 the impact of both regularization terms on the training process of a dense network.

Different targets. We notice stark differences in outcomes when regularizing for the different targets. Where the classic weight decay significantly hinders the convergence of the network once λ increases, choosing the initialization as target suffers significantly less. If we take a look then at the Learning distance, we see that – predictably – optimizing for $||\theta - \theta_0||_2$ finds a solution closer to the initialization, while for weight decay the learning distance converges to $||\theta_0||_2$ (113.05 for ResNet-34, see $\lambda = 10^{-2}$). By comparing these results with Tables 1 and 3, it becomes evident that just because the inferior hyperparameter set can't find good solutions close to initialization, that doesn't mean those don't exist.

Explicit optimization in Phase II. Next we study the impact of this explicit optimization in the *Mask Search* phase, comparing again optimizing for $||\theta||_2$ vs $||\theta - \theta_0||_2$ (Table 5). Here we can clearly see the difference in behavior between weight decay and learning distance minimization. While we can see that accuracy degradation becomes less severe for learning distance minimization when increasing λ, the same property does not hold for weight decay. In that case, progress is more erratic, with no real clear trend. We do however notice that at some sparsity model capacity generally breaks down no matter what configuration, as we see a much less pronounced difference – except for more frequent failure cases with Weight Decay – at 98.85% sparsity.

4.3 Finding Lottery Tickets efficiently.

One of the main critiques of the Lottery Ticket Hypothesis is that finding a sparse ticket is slow and inefficient due to the repeated training and rewinding in combination with a low pruning rate. Having

Table 5: How different types and strengths of regularization impact a Lottery Ticket's trainability.

	$\|\|M^i \odot \theta^i\|\|_2$				$\|\|M^i \odot \theta^i - M^i \odot \theta_0\|\|_2$		
Sparsity	$\lambda = 0.0$	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
67.23%	-0.84%	-1.28%	-1.06%	–	-0.96%	**+0.23%**	+0.03%
89.26%	-3.09%	-27.79%†	-3.54%	–	-3.22%	-1.46%	-2.09%
96.48%	-6.11%	-28.05%†	-8.33%	–	-5.00%	-4.65%	-3.29%
98.85%	-10.68%	-48.65%†	-63.24%†	–	-10.62%	-8.95%	-10.43%
Dense	63.74%						

Table 6: Accuracy and cost proportional to the baseline configuration to find a 96.48% sparse network with different **Phase II** recipes.

Phase II recipe					
T	λ	k	Accuracy	Cost (eps, agg.)	Speedup
200eps	-	0.20	35.69%†	3000	1×
20eps	-	0.20	58.54%	300	10×
20eps	10^-2	0.20	59.70%	300	10×
20eps	10^-2	0.488	60.56%	100	30×
20eps	10^-2	0.6723	59.92%	60	50×
Permuted Baseline			36.29%†	/	-
Randomly pruned			58.06%	/	-

demonstrated in Table 2 that reducing the training budget in **Phase II** can already have beneficial effects compared to using a full budget, we next push this further by varying the pruning rate k, and combining the regularization with the limited budget. We show the results of this exploration for extracting a 96.48% sparse ticket in Table 6. In the most efficient scenario, by combining a 20 epochs budget, severe regularization, and a pruning rate which prunes as much as 5 rounds of $k = 0.2$, we actually outperform the baseline approach, while being 50 times faster. Additional evidence of the weakness of the IMP baseline can be seen when comparing it to a layerwise-permuted ticket, and a one-shot randomly pruned network. We see in both cases that the IMP baseline performs worse at this sparsity, indicating that the mask found by IMP in those cases is actively harmful to generalization.

5 Analysis

5.1 Relation with sign flips

The importance of parameter signs. Gadhikar & Burkholz (2024) posits that Learning Rate Rewinding (LRR) (Renda et al., 2020) outperforms IMP because it can recover from bad parameter signs in the initialization. Essentially when rewinding, LRR preserves the signs after convergence, while IMP reverts them to their initial state. If the network grows less overparameterized, IMP struggles to flip them to beneficial states, leading to accuracy degradation, while LRR can restart from more optimal signs. Our experiments show that the IMP solutions which are consistently more trainable actually have fewer sign flips during dense training (Table 7) due to finding solutions closer to the initialization. This demonstrates that leaning into the purported weakness of IMP can be a valid alternative to facilitating sign flips in underparameterized settings (as in (Gadhikar et al., 2025)).

5.2 Analogy with pretraining

Pretraining as learning distance reduction. One of the effects of pretraining, other than encoding some partial generalization in the model, is that it also pushes the rewind point closer to the trained solution. In that aspect, it functions similarly to the *Mask Search* improvements we have explored in

Table 7: Sign flips during dense training for **(a)** different hyperparameters, and **(b)** different regularization strengths.

	$\mu = 0.0$		$\mu = 0.9$	
	$b = 100$	$b = 256$	$b = 100$	$b = 256$
$T = 20$eps	18.40%	**11.23%**	44.88%	31.27%
$T = 200$eps	45.68%	36.87%	49.81%	49.74%

(a)

	$\|\|M_i \odot \theta_i\|\|_2$	$\|\|M^i \odot \theta^i - M^i \odot \theta_0\|\|_2$
$\lambda = 0$	44.84%	44.84%
$\lambda = 10^{-4}$	**49.81%**	30.85%
$\lambda = 10^{-3}$	49.73%	13.01%
$\lambda = 10^{-2}$	34.69%†	**4.08%**

(b)

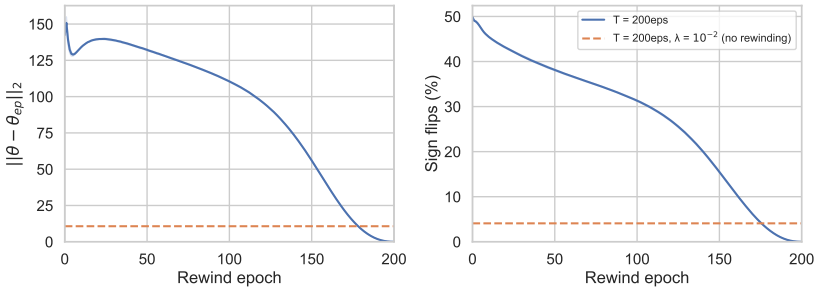


Figure 2: The influence of pretraining on $||\theta - \theta_{\text{ep}}||_2$, and sign flips.

this paper. Concretely, we can see in Figure 2 where we plot the distance and sign flips to the *trained solution*, that choosing a later rewind point can also reduce learning distance and number of flips. This seemingly points to a similar mechanism for which pretraining reduces accuracy degradation while pruning, with the added bonus of a better generalizing starting point. The distinct advantage of our tested approaches over it is that the aspect of determining the correct amount of pretraining is often done by calculating the linear mode connectivity which is an expensive operation, while our approach works out-of-the-box.

5.3 Emergent features

What is hidden in a sparse mask? We have seen that some tickets degrade significantly more than others at high sparsities, leading us to question what the difference is between different masks. For this, we forego the *Sparse Training* phase completely, and employ a probing technique. Earlier work by Zhou et al. (2019) proved the existence of supermasks, which are tickets that can achieve better than random chance predictions without any training. While these have been found in very simple settings with a different pruning criterion, we were unable to replicate these results on more complex settings. Instead we consider a less powerful phenomenon. Rather than keeping both feature extractor and classification layer fixed at initialization, instead we only keep the feature extractor frozen, and train a linear layer on top of the features in the network, following the approach of (Alain & Bengio, 2017). To put the feature quality into context, we compare additionally with a permuted ticket, where the sparsity masks are permuted in a layerwise fashion, thus having the same layerwise sparsity and initialization.

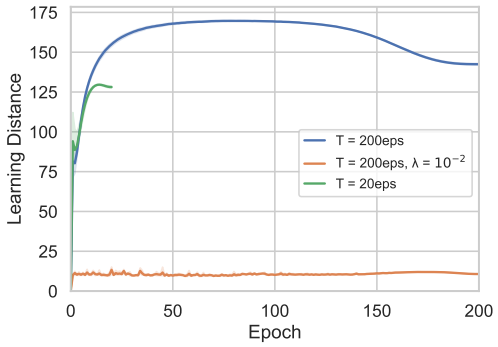


Figure 1: Learning Distance during training in different settings.

Table 8: Validation accuracy for a linear classifier trained on *frozen* ResNet-34 lottery tickets found with various **Phase II** hyperparameters.

Hyperparameters			Sparsities			
μ	b	T	67.23%	89.26%	96.48%	98.85%
0.0	100	20eps	$7.51 \pm 0.49\%$	$9.67 \pm 0.23\%$	$9.58 \pm 0.22\%$	$8.28 \pm 0.42\%$
0.0	256	20eps	$\mathbf{8.87 \pm 0.42\%}$	$\mathbf{10.11 \pm 0.15\%}$	$\mathbf{10.55 \pm 0.25\%}$	$\mathbf{9.75 \pm 0.75\%}$
0.9	100	20eps	$3.89 \pm 0.10\%$	$3.60 \pm 0.24\%$	$3.42 \pm 0.19\%$	$3.46 \pm 0.48\%$
0.9	256	20eps	$3.84 \pm 0.28\%$	$4.34 \pm 0.34\%$	$4.06 \pm 0.31\%$	$3.82 \pm 0.14\%$
0.0	100	200eps	$3.83 \pm 0.19\%$	$3.84 \pm 0.27\%$	$3.85 \pm 0.37\%$	$3.36 \pm 0.12\%$
0.0	256	200eps	$\mathbf{4.53 \pm 0.47\%}$	$\mathbf{5.53 \pm 0.70\%}$	$\mathbf{5.02 \pm 0.32\%}$	$\mathbf{4.29 \pm 0.09\%}$
0.9	100	200eps	$1.89 \pm 0.27\%$	$1.59 \pm 0.40\%$	$1.76 \pm 0.50\%$	$1.65 \pm 0.21\%$
0.9	256	200eps	$2.63 \pm 0.53\%$	$2.04 \pm 0.90\%$	$1.93 \pm 0.44\%$	$1.97 \pm 0.37\%$
$\lambda = 10^{-4}$			$3.17 \pm 0.23\%$	$3.39 \pm 0.62\%$	$3.36 \pm 0.66\%$	$3.62 \pm 0.05\%$
$\lambda = 10^{-3}$			$3.03 \pm 0.39\%$	$7.22 \pm 2.37\%$	$9.34 \pm 1.59\%$	$9.27 \pm 1.71\%$
$\lambda = 10^{-2}$			$3.88 \pm 0.19\%$	$5.59 \pm 0.70\%$	$9.67 \pm 0.50\%$	$11.54 \pm 0.52\%$
Permuted Ticket			$3.39 \pm 0.09\%$	$4.09 \pm 0.15\%$	$0.50 \pm 0.00\%$	$0.50 \pm 0.00\%$
Dense Baseline			$3.13 \pm 0.45\%$			

Closeness to initialization invokes useful features. When probing the different untrained tickets (Table 8) we notice that within tickets found closer to the initialization, the combination of the random weights with the sparsity mask encodes useful features, significantly outperforming random predictions (=0.5%). This indicates iterative pruning can act as some form of feature selection when the intermediate networks are close to the initialization. Sparsity of the network has a small part in this property, as validation accuracy is increasing with increasing sparsity for most configurations. This further hints at the feature selection property of IMP, where the iterative process continues shaping the optimal sparsity mask. In the appendix in Section F, we furthermore show that this presence of useful features at initialization can improve few-shot learning capabilities, as well as that these features at initialization are transferable when frozen to other datasets.

6 Conclusion

We have demonstrated that it is possible to skip the *Pretraining* phase and still find lottery tickets without significant accuracy degradation. To accomplish this, we first explored a variety of hyperparameter combinations in the *Mask Search* phase, and found that settings which were beneficial to dense network generalization negatively impact the trainability of the resulting sparse network. This means that purely by smart hyperparameter selection, the final ticket accuracy improves vastly.

To determine the practical effect of different hyperparameters, we turn towards the concept of learning distance, as measured by the L2-distance from the initialization. While this notion is conceptually very similar to the technique of weight decay, their applications have vastly different impacts on both generalization of dense networks, and trainability of the resulting sparse networks. In particular, we can notice that a lower learning distance results in better trainable sparse networks, both as an implicit result of the different hyperparameter configurations, but also as an explicit result when employing it with regularization. By limiting learning distance, we also limit the number of diverging parameter signs between the initialization and dense network, which solves one of the weaknesses of IMP, namely that it cannot recover as well from such sign flips (Gadhikar & Burkholz, 2024).

To further highlight the impact of our modifications, we conduct an ablation test with the goal of limiting the *Mask Search* duration without impacting generalization. We show that, by combining more efficient hyperparameters with regularization, this overhead can be reduced significantly without impacting the resulting generalization. Finally, we show via a linear probing technique that the masks produced via techniques that limit the learning distance during *Mask Search* can contain useful features at initialization. This means that under the right circumstances IMP can serve as a feature selection technique, which is not observed under common settings.

AI USE STATEMENT

In this work we did not use generative AI tools for the main scientific contributions, such as generating hypotheses, designing or providing feedback on experiments and interpreting the results. This work relies entirely on publicly available datasets. Minor proofreading has been done for clarity and structure by genAI tools, but this feedback was judged manually by the authors for applicability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We share our anonymized codebase (excluding publicly available datasets) in the following repository: `https://anonymous.4open.science/r/iclr_lth_distance-EB6C/`.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations Workshops*, 2017.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. doi: 10.1109/CVPR46437.2021.01604.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Shrey Desai, Hongyuan Zhan, and Ahmed Aly. Evaluating lottery tickets under distributional shifts. In *DeepLo@EMNLP-IJCNLP 2019 - Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource Natural Language Processing - Proceedings*, 2021. doi: 10.18653/v1/d19-6117.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pp. 1309–1318, 2018.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. pp. 3259–3269. PMLR, 2020a. ISBN 2640-3498.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020b.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021.

Advait Gadhikar, Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Sign-in to the lottery: Reparameterizing sparse training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Advait Harshal Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. In *The Twelfth International Conference on Learning Representations*, 2024.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pp. 181–195. Springer, 2022.

Zheng He, Zeke Xie, Quanzhi Zhu, and Zengchang Qin. Sparse double descent: Where network pruning aggravates overfitting. In *International Conference on Machine Learning*, pp. 8635–8659. PMLR, 2022.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017.

Stanislaw Jastrzebksi, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Samy Jelassi and Yuanzhi Li. Towards understanding how momentum improves generalization in deep learning. In *International Conference on Machine Learning*, pp. 9965–10040. PMLR, 2022.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

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

Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

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

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International conference on artificial intelligence and statistics*, pp. 4313–4324. PMLR, 2020.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *International Conference on Learning Representations*, 2022.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

Jaron Maene, Mingxiao Li, and Marie-Francine Moens. Towards understanding iterative magnitude pruning: Why lottery tickets win. *arXiv preprint arXiv:2106.06955*, 2021.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 6682–6691. PMLR, 3 2020.

Rahul Mehta. Sparse transfer learning via winning lottery tickets. *arXiv preprint arXiv:1905.07785*, 2019.

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.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

Mansheej Paul, Brett Larsen, Surya Ganguli, Jonathan Frankle, and Gintare Karolina Dziugaite. Lottery tickets on a data diet: Finding initializations with sparse trainable networks. *Advances in Neural Information Processing Systems*, 35:18916–18928, 2022.

Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? In *The Eleventh International Conference on Learning Representations*, 2023.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.01191.

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

Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D. Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020.

Marc Vischer, Robert Tjarko Lange, and Henning Sprekeler. On lottery tickets and minimal task representations in deep reinforcement learning. In *International Conference on Learning Representations*, 2022.

LI Xuhong, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International conference on machine learning*, pp. 2825–2834. PMLR, 2018.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*, 2020.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Marco Zullich, Eric Medvet, Felice Andrea Pellegrino, and Alessio Ansuini. Speeding-up pruning for artificial neural networks: introducing accelerated iterative magnitude pruning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 3868–3875. IEEE, 2021.

A Appendix

In this appendix, we provide the following additional resources:

- A full algorithmic overview of the Lottery Ticket Hypothesis, and Iterative Magnitude Pruning (Section B).
- In Section C we show hyperparameter results in Phase II for different dataset-network combinations, extended by showing a small comparison with pretrained models. We preface this with a more in-depth study of the impact of Phase II and Phase III hyperparameters. Finally, this section also shows a more thorough hyperparameter analysis, covering more extreme batch sizes, more granular momentum values and learning rate scaling.
- Section D extends the learning distance analysis to the other dataset-network combinations and hyperparameter configurations, including the impact of learning rate scaling and weight decay, and evaluates learning distance regularization on ResNet-18 + CIFAR-10 and VGG-16 + CIFAR-100.
- The sign flips w.r.t. the initialization for the other dataset-network combinations are listed in Section E.
- In Section F we show that useful features also emerge in untrained tickets of the other settings. As an additional analysis, we show that these features transfer to other datasets, and that they improve few-shot generalization.
- A full default hyperparameter configuration is listed in Section G.

B Full Iterative Magnitude Pruning Algorithm

Algorithm 1 Iterative Magnitude Pruning

Given: A neural network initialized with weights $\theta_0 \in \mathbb{R}^d$, a pretraining budget of p epochs, a training budget T, a number of pretraining iterations N, and a pruning rate k.
Initialize a binary pruning mask $M_0 = \mathbb{1}^d$.
Train θ_0 for p epochs to θ_{pt}. ▷ *Pretraining*
for level $L \in \{1, \ldots, N\}$ **do** ▷ *Mask Search*
    Let $M_{L-1} \odot \theta_L$ be the result of training $M_{L-1} \odot \theta_{pt}$ for $T - p$ epochs.
    Initialize M_L as M_{L-1}.
    Let $M_L[i] = 0$ for the $k\%$ lowest magnitude connections in $M_{L-1} \odot \theta_L$.
end for
Train the lottery ticket $M_N \odot \theta_{pt}$ to completion. ▷ *Sparse Training*

C Hyperparameter selection in **Phase II**

Setup. Rather than employing a fixed hyperparameter setting in the *Sparse Training* phase, as used in the main paper, instead we here use the same **Phase II** and **Phase III** hyperparameters. This means that typically there is some room for improvement for the tickets that are trained without momentum, or with a higher batch size, f.e. compare the values in Table 1 with Table 11. However, we first show in a disentangling experiment that no matter what hyperparameter values are used for the *Sparse Training* phase, if those are fixed, our observations of the main paper on hyperparameters in the *Mask Search* phase hold.

C.1 Disentangling Mask Search and Sparse Training

Generalization is primarily determined by Mask Search hyperparameters. In the tables below (Tables 9 to 11, we split the μ and b values of the both phases, in an attempt to disentangle the impact of *Mask Search* and *Sparse Training*. Symbols are annotated with their respective phase via the subscripts $\texttt{MS}$, $\texttt{ST}$ for *Mask Search*, respectively *Sparse Training*. We can see that – for each combination of $\mu_{\text{ST}}, b_{\text{ST}}$ – tickets found with $\mu_{\text{MS}} = 0.0, b_{\text{MS}} = 256$ outperform those found with

$\mu_{\text{MS}} = 0.9, b_{\text{MS}} = 100$ when each is trained with the same $\mu_{\text{ST}}, b_{\text{ST}}$. This leads us to believe that the primary driver behind the generalization of a lottery ticket is the structure of the sparsity mask, which is determined by the hyperparameters used in the Mask Search phase.

Table 9: Validation accuracy for **ResNet-18** Tickets on **CIFAR-10** for different combinations of Mask Search (μ_{MS}, b_{MS}) and Sparse Training (μ_{ST}, b_{ST}) hyperparameters.

Hyperparameters				Sparsity				
μ_{MS}	b_{MS}	μ_{ST}	b_{ST}	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	256	0.0	100	94.58%	94.55%	94.37%	**93.79%**	**92.80%**
		0.0	256	93.63%	94.02%	93.93%	93.39%	92.67%
		0.9	100	95.24%	**95.11%**	**94.78%**	93.68%	91.82%
		0.9	256	95.18%	94.99%	94.56%	93.55%	92.05%
0.9	100	0.0	100	94.58%	94.05%	93.14%	91.99%	90.06%
		0.0	256	93.63%	92.85%	92.30%	90.79%	89.07%
		0.9	100	95.24%	**94.99%**	**94.39%**	**93.25%**	**91.41%**
		0.9	256	95.18%	94.72%	94.13%	93.16%	91.22%

Table 10: Validation accuracy for **VGG16** Tickets on **CIFAR-100** for different combinations of Mask Search (μ_{MS}, b_{MS}) and Sparse Training (μ_{ST}, b_{ST}) hyperparameters.

Hyperparameters				Sparsity				
μ_{MS}	b_{MS}	μ_{ST}	b_{ST}	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	256	0.0	100	72.77%	70.27%	69.11%	67.74%	**65.98%**
		0.0	256	69.26%	69.53%	68.80%	67.54%	65.41%
		0.9	100	73.90%	**73.30%**	**71.73%**	**69.29%**	62.79%
		0.9	256	**73.98%**	72.57%	70.19%	68.60%	65.43%
0.9	100	0.0	100	72.77%	71.61%	69.69%	66.72%	56.62%
		0.0	256	69.26%	68.72%	66.55%	63.48%	52.98%
		0.9	100	73.90%	73.45%	72.31%	68.19%	55.91%
		0.9	256	**73.98%**	**73.85%**	**72.28%**	**68.96%**	**58.72%**

Table 11: Validation accuracy for **ResNet-34** Tickets on **TinyImageNet** for different combinations of Mask Search (μ_{MS}, b_{MS}) and Sparse Training (μ_{ST}, b_{ST}) hyperparameters. Entries annotated with † encompass one or multiple runs in which the ticket did not converge and remained at random chance.

Hyperparameters				Sparsity				
μ_{MS}	b_{MS}	μ_{ST}	b_{ST}	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	256	0.0	100	61.14%	60.18%	60.60%	58.98%	55.94%
		0.0	256	59.33%	59.95%	60.20%	58.89%	**57.09%**
		0.9	100	**63.74%**	**62.83%**	**61.56%**	**59.48%**	55.27%
		0.9	256	61.57%	60.78%	59.49%	57.27%	54.45%
0.9	100	0.0	100	61.14%	58.48%	51.55%	48.85%	42.70%
		0.0	256	59.33%	56.73%	49.74%	47.09%	41.34%
		0.9	100	**63.74%**	**62.46%**	35.95%†	35.69%†	15.09%†
		0.9	256	61.57%	59.28%	**52.33%**	**49.58%**	**44.32%**

C.2 FULL RESULTS

In this section, we give both visual (Figures 3 to 8) and tabular (Tables 12 to 16) representations of the lottery ticket performances at different sparsities. **Note that these results use the same Phase II and III hyperparameters, as such they are not fully comparable with the main paper results!** For completeness, we also compare with a 1% pretrained network, to more accurately gauge the impact of pretraining (indicated by Pretrained / Late-rewinding in the figures). We can see that simply adding pre-training to the $\{\mu = 0.9, b = 100\}$ case does not necessarily reduce the accuracy degradation, indeed, the best hyperparameters from Table 2 without regularization can still outperform those pretrained tickets. That being said, if hyperparameters are fixed, then pretraining universally improves performance, which is to be expected.

VGG-16 divergence. For the results found with VGG-16 we notice that the inflection point where the more stable hyperparameters start to outperform unstable hyperparameters occurs at a higher sparsity than for the ResNet models. Likely this is either a result of the different overparameterization of VGG-16 vs ResNet-18, or related to the use of skip connections. We leave an exact explanation for this phenomenon for future work.

Non-convergence in the graphs. As mentioned in the main paper, ResNet-34 + TinyImageNet results did not always converge. As such, some of the curves record a significant accuracy drop together with a high standard deviation. This indicates that at that iteration one (or more) random runs did not train and instead remained at random chance (0.5%).

Table 12: Lottery Ticket extracted from **ResNet-18** on **CIFAR-10** performances for different training parameters at different sparsity levels. Winning tickets are underlined.

Hyperparameters			Sparsity				
μ	b	Pretrained	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	100	✗	94.58%	94.39%	93.93%	92.95%	91.10%
0.0	256	✗	93.63%	94.02%	93.93%	**93.39%**	**92.67%**
0.9	100	✗	**95.24%**	**94.99%**	**94.39%**	93.25%	91.41%
0.9	256	✗	95.18%	94.98%	94.29%	93.36%	91.39%
0.0	100	✓	94.62%	95.06%	**94.95%**	**94.71%**	**93.95%**
0.0	256	✓	93.54%	93.97%	93.94%	93.50%	92.77%
0.9	100	✓	**95.30%**	95.13%	94.85%	93.90%	92.31%
0.9	256	✓	95.24%	**95.16%**	94.55%	93.57%	91.86%

Table 13: Lottery Ticket extracted from **ResNet-18** on **CIFAR-100** performances for different training parameters at different sparsity levels. Winning tickets are underlined.

Hyperparameters			Sparsity				
μ	b	Pretrained	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	100	✗	76.66%	75.57%	73.53%	70.24%	66.43%
0.0	256	✗	74.39%	75.03%	**74.46%**	**72.43%**	**68.65%**
0.9	100	✗	**78.54%**	**76.60%**	73.91%	70.33%	65.05%
0.9	256	✗	77.74%	76.47%	74.07%	70.23%	65.89%
0.0	100	✓	76.57%	76.76%	76.23%	74.23%	**69.99%**
0.0	256	✓	74.41%	74.62%	73.54%	71.76%	67.93%
0.9	100	✓	**78.18%**	77.20%	74.66%	70.58%	65.80%
0.9	256	✓	77.71%	**77.57%**	**77.24%**	**74.35%**	69.21%

Table 14: Lottery Ticket extracted from **VGG16** on **CIFAR-100** performances for different training parameters at different sparsity levels. Winning tickets are underlined.

Hyperparameters			Sparsity				
μ	b	Pretrained	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	100	✗	72.77%	71.60%	69.81%	66.37%	60.91%
0.0	256	✗	69.26%	69.53%	68.80%	67.54%	**65.41%**
0.9	100	✗	73.90%	73.45%	**72.31%**	68.19%	55.91%
0.9	256	✗	**73.98%**	**73.84%**	72.25%	**69.50%**	62.52%
0.0	100	✓	72.72%	72.83%	72.90%	**71.93%**	64.43%
0.0	256	✓	69.57%	70.14%	69.06%	67.90%	**65.27%**
0.9	100	✓	**74.05%**	73.62%	72.84%	68.83%	53.42%
0.9	256	✓	73.99%	**74.05%**	**73.23%**	70.50%	64.63%

Table 15: Lottery Ticket extracted from **Swin** models on **TinyImageNet** performances for different training parameters at different sparsity levels. Winning tickets are underlined.

Hyperparameters			Sparsity				
μ	b	Pretrained	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	100	✗	50.65%	**52.01%**	**51.01%**	46.07%	45.55%
0.0	256	✗	49.52%	50.01%	48.97%	**48.79%**	**45.84%**
0.9	100	✗	**52.53%**	51.89%	50.02%	45.53%	45.36%
0.9	256	✗	52.37%	50.57%	49.37%	45.76%	45.41%
0.0	100	✓	50.29%	**52.39%**	**51.72%**	49.05%	**46.91%**
0.0	256	✓	49.32%	50.14%	49.02%	**49.90%**	46.26%
0.9	100	✓	52.21%	51.69%	49.95%	46.79%	46.11%
0.9	256	✓	**52.46%**	51.92%	50.19%	47.71%	46.90%

Table 16: Lottery Ticket extracted from **ResNet-50** on **ImageNet-100** performances for different training parameters at different sparsity levels. Winning tickets are underlined.

Hyperparameters			Sparsity			
μ	b	Pretrained	0.00%	67.23%	89.26%	96.48%
0.0	100	✗	82.67%	83.03%	**82.31%**	**81.75%**
0.0	256	✗	80.67%	81.35%	80.60%	80.37%
0.9	100	✗	**84.94%**	**83.93%**	79.57%	73.07%
0.9	256	✗	81.83%	82.26%	81.93%	79.52%
0.0	100	✓	82.83%	83.53%	83.10%	**82.96%**
0.0	256	✓	80.54%	81.17%	81.01%	80.34%
0.9	100	✓	**84.61%**	**84.93%**	**84.09%**	81.11%
0.9	256	✓	81.97%	83.40%	83.39%	81.07%

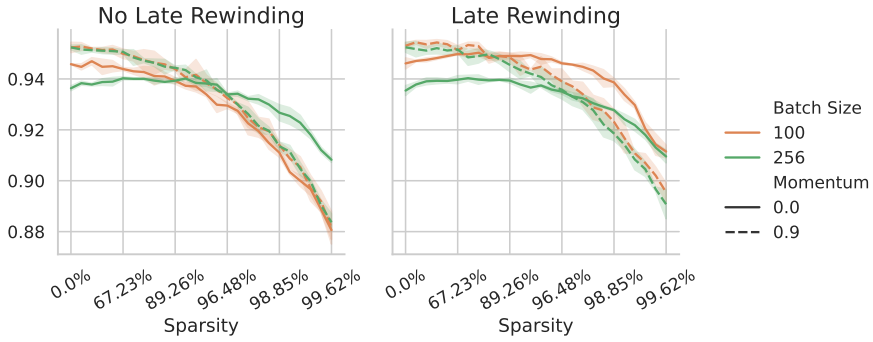


Figure 3: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-18** model on the **CIFAR-10** dataset.

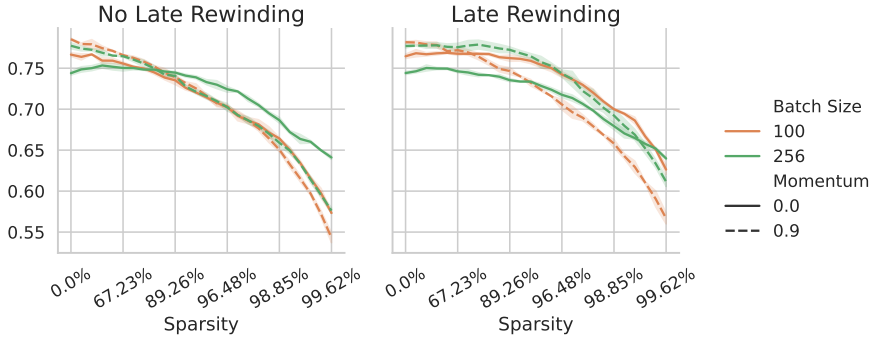


Figure 4: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-18** model on the **CIFAR-100** dataset.

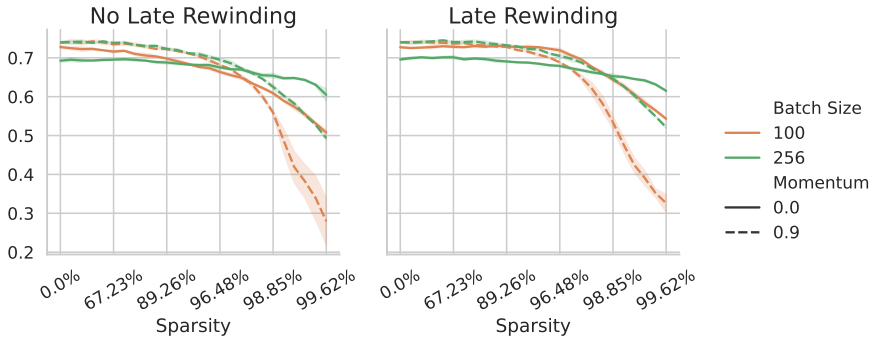


Figure 5: Validation accuracies at different sparsities for Lottery Tickets extracted from a **VGG-16** model on the **CIFAR-100 dataset**.

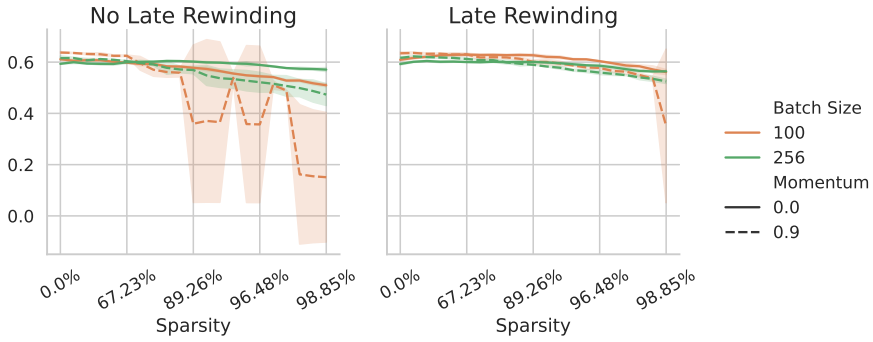


Figure 6: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-34** model on the **TinyImageNet** dataset.

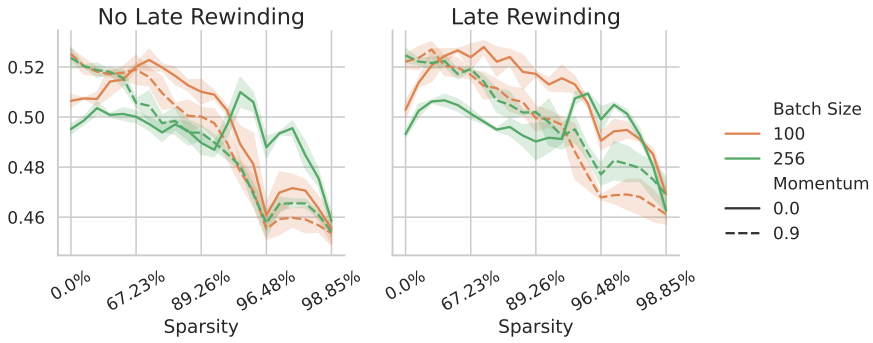


Figure 7: Validation accuracies at different sparsities for Lottery Tickets extracted from a **Swin** model on the **TinyImageNet** dataset.

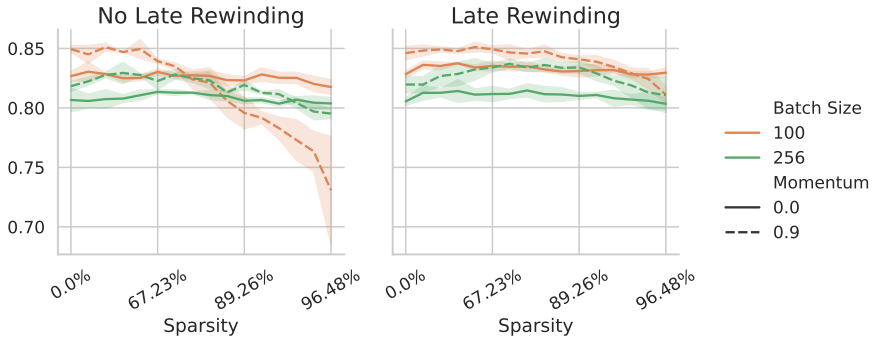


Figure 8: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-50** model on the **ImageNet-100** dataset.

C.3 More hyperparameter values

Different batch sizes. In Table 17, we extend the range of batch sizes tested for ResNet-18 + CIFAR-10 including batch sizes which are typically not used in practice. More specifically, this means using very low batch sizes, such as [8, 32] or very high batch sizes such as [1024, 2048]. We show that smaller batch sizes suffer from significant degradation at higher sparsities, but have higher accuracy at the earlier sparsities.

Table 17: Validation accuracy of **ResNet-18** lottery tickets extracted for **CIFAR-10** given more extreme batch sizes at different sparsity levels.

Hyperparameters		Sparsity				
μ	b	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	8	$95.29 \pm 0.00\%$	$\mathbf{95.54 \pm 0.00\%}$	$\mathbf{94.81 \pm 0.00\%}$	$\mathbf{93.79 \pm 0.00\%}$	$\mathbf{91.93 \pm 0.00\%}$
	32	$\mathbf{95.41 \pm 0.00\%}$	$95.29 \pm 0.00\%$	$94.71 \pm 0.00\%$	$93.68 \pm 0.00\%$	$91.72 \pm 0.00\%$
	1024	$91.87 \pm 0.00\%$	$91.81 \pm 0.00\%$	$91.62 \pm 0.00\%$	$91.24 \pm 0.00\%$	$89.86 \pm 0.00\%$
	2048	$90.92 \pm 0.00\%$	$90.74 \pm 0.00\%$	$90.37 \pm 0.00\%$	$90.00 \pm 0.00\%$	$88.78 \pm 0.00\%$
0.9	8	$92.92 \pm 0.00\%$	$88.31 \pm 0.00\%$	$77.92 \pm 0.00\%$	$70.27 \pm 0.00\%$	$10.00 \pm 0.00\%$
	32	$\mathbf{94.53 \pm 0.00\%}$	$93.75 \pm 0.00\%$	$91.49 \pm 0.00\%$	$90.02 \pm 0.00\%$	$88.62 \pm 0.00\%$
	1024	$94.03 \pm 0.00\%$	$\mathbf{94.30 \pm 0.00\%}$	$93.54 \pm 0.00\%$	$92.80 \pm 0.00\%$	$90.57 \pm 0.00\%$
	2048	$93.14 \pm 0.00\%$	$93.46 \pm 0.00\%$	$\mathbf{93.60 \pm 0.00\%}$	$\mathbf{93.54 \pm 0.00\%}$	$\mathbf{92.31 \pm 0.00\%}$

Different momentum values. While it is most common to use $\mu = 0.9$ or $\mu = 0.0$ in training configurations for neural networks, we will explore additional values for this momentum parameter in this section. With this we can provide a more granular overview of the effects on the validation accuracy of the resulting lottery tickets. By comparing the granular momentum results from Table 18 with those found using the common values of μ in Table 12, we notice that our observations in the main paper also holds true for the more granular values of μ. Indeed, solutions found with a lower value of μ exhibit lower validation accuracy for the dense network, but have a less severe accuracy drop than those found with higher values of μ. Interestingly the results with $\mu = 0.3$ slightly outperform the results with $\mu = 0.0$, indicating that some small amount of momentum can be beneficial.

Table 18: Validation accuracy of **ResNet-18** lottery tickets extracted for **CIFAR-10** with different values for μ at different sparsity levels.

Hyperparameters		Sparsity				
b	μ	0.00%	67.23%	89.26%	96.48%	98.85%
256	0.3	$93.95 \pm 0.36\%$	$93.92 \pm 0.08\%$	$\mathbf{94.07 \pm 0.23\%}$	$\mathbf{93.92 \pm 0.27\%}$	$\mathbf{92.96 \pm 0.17\%}$
	0.6	$94.49 \pm 0.23\%$	$\mathbf{94.31 \pm 0.16\%}$	$93.76 \pm 0.30\%$	$92.82 \pm 0.14\%$	$90.86 \pm 0.17\%$
	0.95	$\mathbf{95.15 \pm 0.11\%}$	$94.29 \pm 0.06\%$	$93.21 \pm 0.89\%$	$91.58 \pm 1.12\%$	$89.89 \pm 1.34\%$

Learning Rate Scaling. In an optimization problem the learning rate determines (together with the magnitude of the gradient) the step size the optimizer applies for each update of the weights. Typically a larger learning rate is employed early during training to move towards a low-loss region, and then the optimizer will use a lower learning rate to converge within that loss basin. In this paper we employ cosine annealing for this purpose. However, as the same learning rate is used for all experiments, settings with a lower batch size have more weight updates and thus have more flexibility. In Table 19 we employ the linear scaling rule from (Goyal et al., 2017), where multiplying the batch size by a factor k necessitates multiplying the learning rate by the same factor k, to accommodate this and transform from one setting to another. We scale in both directions, defining $\{b = 100, lr = 0.0782\}$ for comparison with $\{b = 256, lr = 0.2\}$ and defining $\{b = 256, lr = 0.512\}$ for comparison with $\{b = 100, lr = 0.2\}$.

We observe that applying this linear scaling rule does not result in comparable validation accuracy with the original settings. Specifically, tickets found with $\{b = 256, lr = 0.512\}$ generalize significantly worse compared to $\{b = 100, lr = 0.2\}$, while the opposite is true for those found with $\{b = 100, lr = 0.0782\}$. This translates into an inversion of the previous trends, where we can now find winning tickets for lower batch sizes, while employing higher batch sizes results in significant accuracy degradation. Importantly, on an even playing field (i.e., if the linear scaling rule is applied in either direction) configurations with $b = 100$ consistently outperform those with $b = 256$.

Table 19: The impact of learning rate on the existence of winning tickets for **ResNet-34** trained on **TinyImageNet**. Entries annotated with † encompass one or multiple runs in which the ticket did not converge and remained at random chance.

Hyperparameters			Sparsity				
μ	b	lr	0.00%	67.23%	89.26%	96.48%	98.85%
0.0	100	0.2	61.14%	59.76%	57.79%	54.47%	50.98%
		0.0782	60.27%	61.05%	61.10%	59.54%	57.79%
0.0	256	0.2	59.33%	59.95%	60.20%	58.89%	57.09%
		0.512	58.50%	57.96%	56.50%	53.46%	49.88%
0.9	100	0.2	**63.74%**	**62.46%**	35.95%†	35.69%†	15.09%†
		0.0782	62.40%	61.43%	58.71%	55.57%	50.45%
0.9	256	0.2	61.57%	60.37%	56.93%	52.76%	47.34%
		0.512	62.48%	59.93%	17.62%†	34.85%†	13.83%†

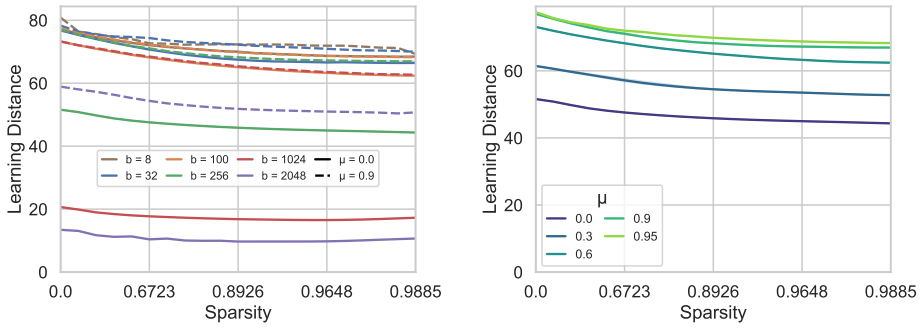
D SGD NOISE AND LEARNING DISTANCE

D.1 HYPERPARAMETERS AND LEARNING DISTANCE

In Figure 12 we show the learning distance for the remaining dataset and network combinations. To further highlight the impact of the hyperparameters on this metric, we also show in Figure 9 the learning distances for the extended hyperparameter configurations. We also show the impact of lr scaling and weight decay on the learning distance in Figures 10 and 11 respectively.

Lr impact. We can clearly see in Figure 10 that by applying lr rescaling, the learning distance is modified significantly. Indeed, when scaling the lr for settings with $b = 100$, the resulting learning distance aligns almost perfectly for settings with $\{b = 256, lr = 0.2\}$ and vice-versa. This indicates that the interplay of learning rate and batch size determines the distance from the initialization, and helps explain the results listed in Table 19.

Weight decay impact. Weight decay imposes an L2 regularization on the weights, preventing them from growing too large, effectively serving as a different kind of limit on the learning distance. While this results in solutions found significantly closer to the initialization (see Figure 11), its application significantly hinders the generalization of sparse tickets.



(a) (b)

Figure 9: Learning distance in the *Mask Search* phase for **ResNet-18** on **CIFAR10** for more diverse **(a)** Batch Sizes and **(b)** Momentum values.

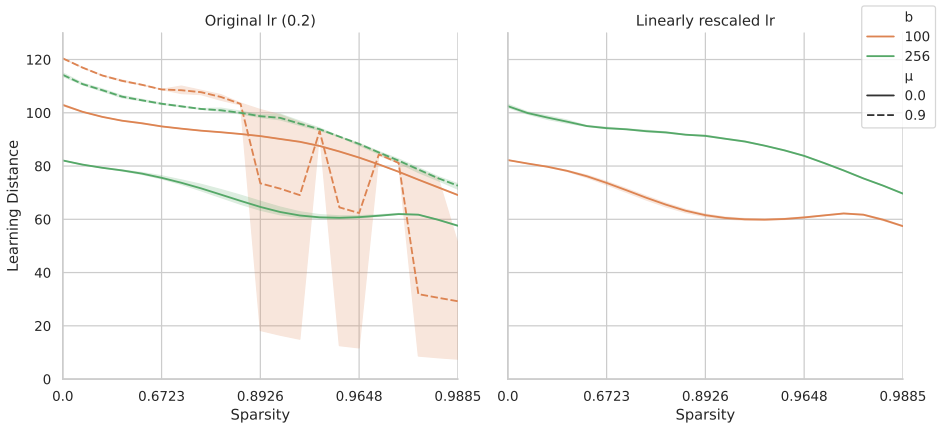


Figure 10: Impact of lr rescaling on the learning distance in the *Mask Search* phase for **ResNet-34** on **TinyImageNet**.

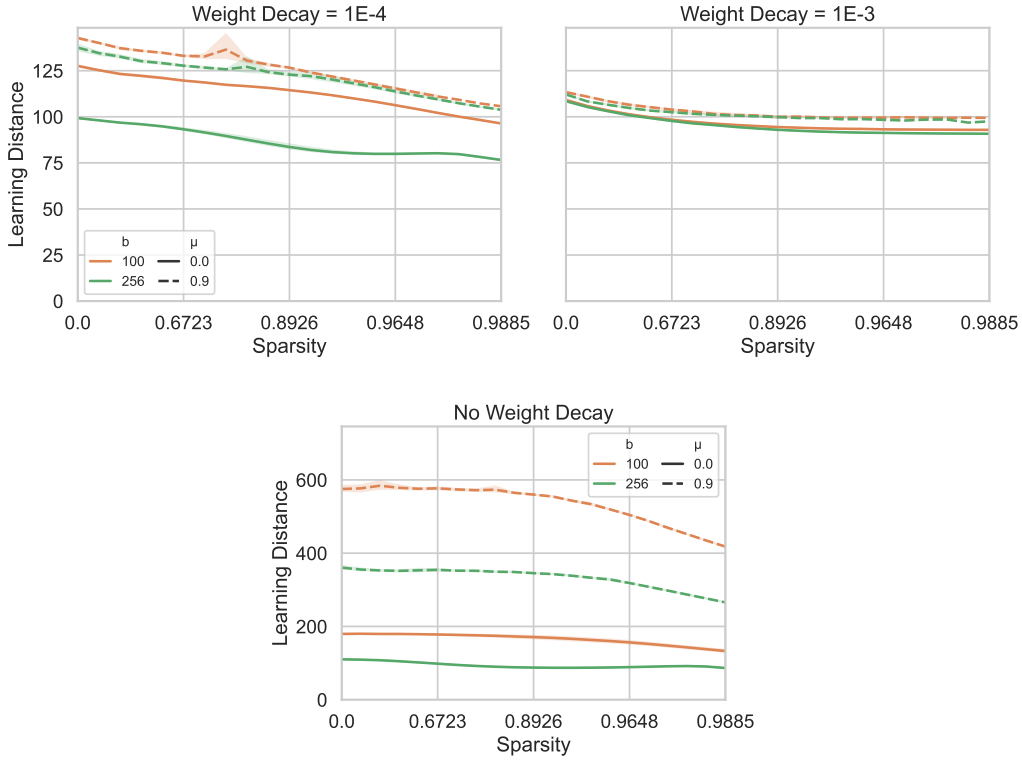
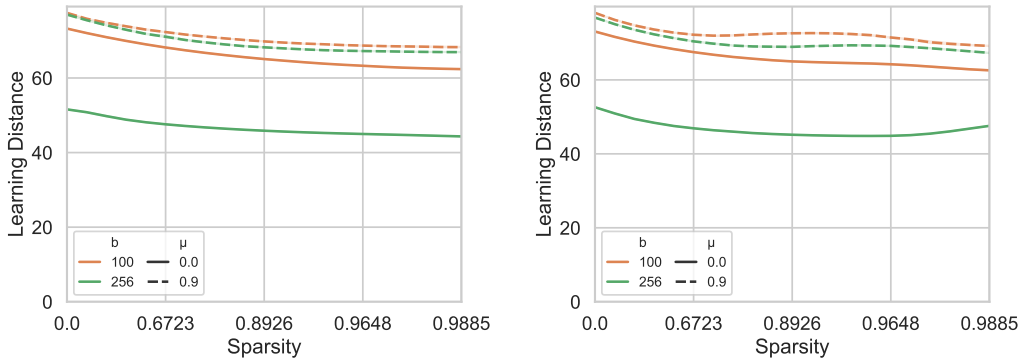
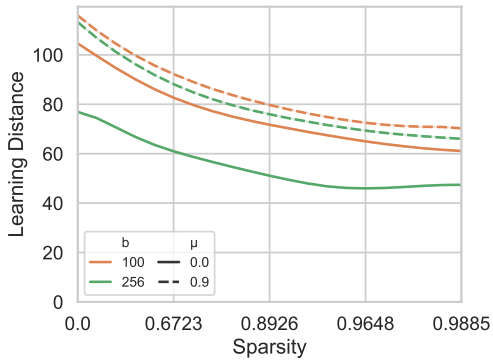


Figure 11: Impact of weight decay on the learning distance in the *Mask Search* phase for **ResNet-34** on **TinyImageNet**.

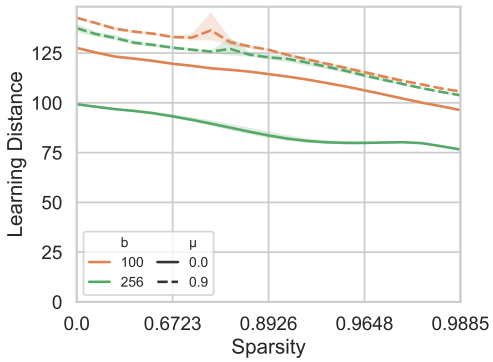


(a) ResNet-18 + CIFAR10

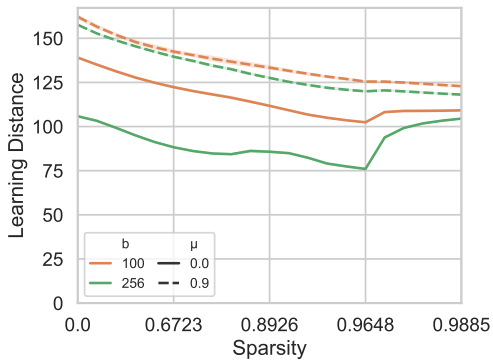
(b) ResNet-18 + CIFAR100



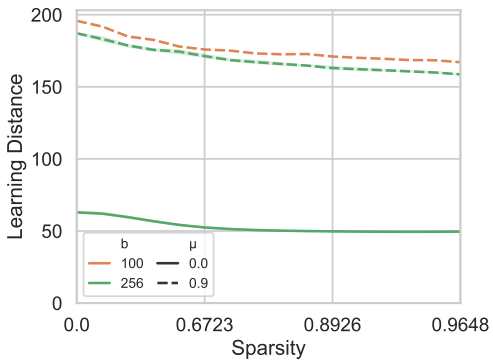
(c) VGG16 + CIFAR100



(d) ResNet-34 + TinyImageNet



(e) SWIN + TinyImageNet



(f) ResNet-50 + ImageNet100

Figure 12: Learning distance in the *Mask Search* phase given different hyperparameter settings for various dataset & network combinations without pretraining.

D.2 Learning Distance Regularization

Below in Table 20 we extend the analysis on the explicit Learning Distance regularization in the main paper on ResNet-18 + CIFAR-10, and VGG-16 + CIFAR-100. We can clearly see that in the case of ResNet-18 on CIFAR-10, explicit regularization actually hinders the generalization of the model, opposite to what we see for the other models. We link this to the complexity of the setting, where we can see that in more complex settings (ResNet-34 + TinyImageNet, higher sparsity entries of VGG16 + CIFAR-100) the advantage of Learning Distance regularization on the trainability of

the lottery ticket is clearly proven. However, a too high regularization can also hinder the trainability of ticket, so depending on the complexity of the task, λ needs to be set higher or lower.

Table 20: How different strengths of learning distance regularization impact a Lottery Ticket's trainability for (a) **ResNet-18** on **CIFAR-10**, (b) **VGG-16** on **CIFAR-100**.

	$\|\|M^i \odot \theta^i - M^i \odot \theta_0\|\|_2$		
Sparsity	$\lambda = 10^{-4}$	10^{-3}	10^{-2}
67.23%	-0.39%	-0.85%	-0.53%
89.26%	-1.02%	-1.89%	-0.89%
96.48%	-1.99%	-3.23%	-2.26%
98.85%	-3.87%	-5.68%	-5.59%
Dense	95.24%		

(a) **ResNet-18** on **CIFAR-10**

	$\|\|M^i \odot \theta^i - M^i \odot \theta_0\|\|_2$		
Sparsity	$\lambda = 10^{-4}$	10^{-3}	10^{-2}
67.23%	-0.64%	-1.77%	-1.20%
89.26%	-2.53%	-3.23%	-2.85%
96.48%	-5.19%	-3.79%	-5.15%
98.85%	-12.37%	-8.60%	-11.46%
Dense	73.90%		

(b) **VGG-16** on **CIFAR-100**

E Relation with Sign flips

Calculating the sign flips for the other dataset-network configurations inTable 21 further reinforces the observation that using $\{\mu = 0.0, b = 256\}$ results in networks with the least sign flips. This lies in line with its lower learning distances in Figure 12.

Table 21: Sign flips w.r.t. the initialization for dense networks under various **Phase II** hyperparameters.

		$\mu = 0.0$		$\mu = 0.9$	
Network	Dataset	$b = 100$	$b = 256$	$b = 100$	$b = 256$
ResNet-18	CIFAR-10	38.20%	**16.01%**	49.86%	49.65%
ResNet-18	CIFAR-100	43.23%	**25.48%**	49.85%	49.55%
VGG-16	CIFAR-100	39.92%	**21.42%**	48.85%	49.39%
Swin	TinyImageNet	40.20%	**30.59%**	47.14%	47.53%
ViT	TinyImageNet	26.22%	**17.80%**	25.96%	32.24%

F Emergent Features

F.1 Other dataset results

Closer tickets encode more useful features. We highlight accuracy for a linear probe trained on the features encoded in a lottery ticket at different sparsities in Tables 22 to 25 for the different dataset-network combinations. We consider the features right before the linear (classification) layer so – depending on the model architecture – these can have different shapes. For the tickets found with the full budget we observe the same results as in the main paper, namely that the closer tickets lead to better generalizable probes. However, when studying tickets found with $T = 20$eps, we notably observe that these hyperparameters are no longer the best performing, which differs from the ResNet-34 + TinyImageNet setting in the main paper.

Absence of Swin results. Interestingly, an outlier is the Swin architecture on TinyImageNet. In that case, the features encoded at initialization in either the dense network or any of the tickets only ever train to random chance, which is why we do not list the results. This is likely due to the bag-of-features approach employed by Transformer models which lack the inherent structural biases of Convolutional Neural Networks (CNNs). We hypothesize that more significant information is encoded in the classification layer, which our approach discards in favor of finetuning one from scratch.

Table 22: **CIFAR-10** validation accuracy for a linear classifier trained on *frozen* **ResNet-18** lottery tickets extracted at different sparsities.

Hyperparameters			Sparsities			
μ	b	Budget	67.23%	89.26%	96.48%	98.85%
0.0	100	20eps	**77.77 ± 1.09%**	**85.85 ± 0.64%**	**85.41 ± 0.56%**	**77.37 ± 0.55%**
0.0	256		68.90 ± 1.27%	80.93 ± 0.68%	80.55 ± 0.99%	68.15 ± 1.80%
0.9	100		28.32 ± 1.10%	30.67 ± 4.08%	29.32 ± 1.22%	32.60 ± 0.19%
0.9	256		43.22 ± 2.19%	51.95 ± 2.99%	55.13 ± 5.37%	46.91 ± 4.23%
0.0	100	200eps	29.54 ± 2.89%	29.54 ± 2.53%	30.84 ± 0.69%	30.96 ± 2.08%
0.0	256		**70.48 ± 1.47%**	**82.87 ± 1.03%**	**80.85 ± 1.93%**	**69.64 ± 2.40%**
0.9	100		30.43 ± 0.78%	28.65 ± 1.13%	25.71 ± 1.27%	22.16 ± 1.39%
0.9	256		27.71 ± 1.58%	27.83 ± 1.90%	28.15 ± 0.25%	28.73 ± 0.45%
$\lambda = 10^{-4}$			30.69 ± 1.88%	30.21 ± 6.39%	31.48 ± 2.51%	31.87 ± 2.31%
$\lambda = 10^{-3}$			30.89 ± 2.94%	32.77 ± 2.15%	29.01 ± 2.55%	31.64 ± 2.57%
$\lambda = 10^{-2}$			30.45 ± 0.82%	41.57 ± 1.52%	56.39 ± 3.90%	53.74 ± 0.54%
Permuted Ticket			30.38 ± 1.62%	33.69 ± 0.56%	31.81 ± 0.61%	10.00 ± 0.00%
Dense Network			28.20 ± 1.52%			

Table 23: **CIFAR-100** validation accuracy for a linear classifier trained on *frozen* **ResNet-18** lottery tickets extracted at different sparsities.

Hyperparameters			Sparsities			
μ	b	Budget	67.23%	89.26%	96.48%	98.85%
0.0	100	200eps	9.94 ± 0.28%	9.31 ± 0.76%	10.51 ± 0.32%	10.70 ± 0.67%
0.0	256		**31.81 ± 1.52%**	**44.26 ± 0.88%**	**36.57 ± 0.87%**	**21.46 ± 0.82%**
0.9	100		7.15 ± 0.37%	7.14 ± 0.27%	7.41 ± 1.54%	8.14 ± 0.68%
0.9	256		7.41 ± 0.44%	7.84 ± 0.08%	8.38 ± 0.62%	8.14 ± 0.68%
Permuted Ticket			8.50 ± 0.18%	10.13 ± 0.61%	4.88 ± 0.18%	1.00 ± 0.00%
Dense Network			8.92 ± 0.99%			

Table 24: **CIFAR-100** validation accuracy for a linear classifier trained on *frozen* **VGG16** lottery tickets extracted at different sparsities.

Hyperparameters			Sparsities			
μ	b	Budget	67.23%	89.26%	96.48%	98.85%
0.0	100	20eps	**25.40 ± 0.75%**	**27.05 ± 1.24%**	**25.62 ± 0.42%**	**23.21 ± 0.50%**
0.0	256		18.30 ± 0.62%	19.43 ± 0.55%	18.65 ± 0.62%	19.80 ± 0.75%
0.9	100		10.24 ± 1.14%	12.01 ± 0.25%	11.21 ± 0.50%	11.20 ± 0.15%
0.9	256		16.23 ± 0.55%	21.47 ± 0.50%	19.88 ± 0.96%	16.20 ± 0.69%
0.0	100	200eps	9.62 ± 0.41%	9.99 ± 0.20%	10.51 ± 0.16%	11.53 ± 0.22%
0.0	256		**16.67 ± 0.47%**	**17.73 ± 1.32%**	**16.71 ± 0.42%**	**12.47 ± 0.37%**
0.9	100		8.53 ± 0.83%	8.33 ± 0.74%	8.49 ± 0.28%	6.19 ± 0.11%
0.9	256		10.46 ± 0.94%	11.39 ± 0.49%	11.27 ± 0.93%	11.41 ± 0.49%
$\lambda = 10^{-4}$			10.26 ± 0.67%	4.43 ± 0.23%	10.88 ± 0.24%	9.87 ± 0.82%
$\lambda = 10^{-3}$			11.43 ± 0.44%	15.31 ± 0.48%	16.60 ± 1.29%	15.49 ± 0.61%
$\lambda = 10^{-2}$			12.64 ± 0.64%	15.07 ± 0.64%	17.12 ± 0.32%	16.66 ± 1.33%
Permuted Ticket			8.57 ± 0.98%	8.41 ± 1.01%	8.56 ± 0.16%	6.13 ± 0.07%
Dense Network			10.88 ± 0.75%			

Table 25: **ImageNet100** validation accuracy for a linear classifier trained on *frozen* **ResNet-50** lottery tickets extracted at different sparsities.

Hyperparameters			Sparsities		
μ	b	Budget	67.23%	89.26%	96.48%
0.0	100	200eps	$8.39 \pm 0.12\%$	$12.71 \pm 0.82\%$	$12.61 \pm 0.20\%$
0.0	256		$\mathbf{11.72 \pm 0.12\%}$	$\mathbf{14.66 \pm 1.01\%}$	$\mathbf{15.09 \pm 0.41\%}$
0.9	100		$9.15 \pm 1.63\%$	$6.64 \pm 0.90\%$	$5.93 \pm 1.35\%$
0.9	256		$9.55 \pm 0.25\%$	$9.59 \pm 0.79\%$	$9.42 \pm 0.21\%$
Permuted Ticket			$8.64 \pm 0.67\%$	$6.77 \pm 1.25\%$	$5.77 \pm 1.24\%$
Dense Network				$9.90 \pm 0.74\%$	

F.2 TRANSFERABILITY

Setup. We ask ourselves whether those features that emerge in the network by pruning are uniquely optimized for the dataset the network is trained on. To evaluate this, we keep the feature extractor of the network frozen, as in the previous experiment, and train a new (dense) linear layer on the final features with the output classes adapted to the new dataset. This incurs a slight loss of sparsity, but is preferable over resparsifying the new classification layer due to computational cost. For transferring, we follow the regime of the *Sparse Training* phase, but use the following datasets: {MNIST, EuroSAT, CIFAR-10, CIFAR-100, CUB-200, TinyImageNet}. For each setting, we adjust for the relative difference in difficulty between source and target dataset. This means that for TinyImageNet we use {EuroSAT, CIFAR-10, CIFAR-100, CUB-200}, while if the source dataset is CIFAR-10 or CIFAR-100, CUB-200 is replaced by the easier MNIST. In the case of ResNet-50 + ImageNet100 we replace EuroSAT with the more complex TinyImageNet. When a size mismatch occurs between the target dataset and the expected input size of the network, we rescale the input to resolve this.

Results. The results for the different datasets and network combinations are listed in Tables 26 to 31. These results follow the trends of the linear evaluation experiments, showing that the features which generalize better to the original dataset, are not specific for that dataset, but also transfer better to other target datasets. In some cases, e.g. transferring from ResNet-18 + CIFAR-10/100 to most datasets, these features can outperform those found in a dense network trained to convergence on the original dataset. The reasoning behind this is not further explored.

Table 26: Transferability to different datasets for frozen 89.26% **ResNet-18** tickets extracted on **CIFAR-10** with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	MNIST	EuroSAT	CIFAR-100	TinyImageNet
0.0	100	20eps	$\mathbf{97.92 \pm 0.08\%}$	$\mathbf{84.62 \pm 0.71\%}$	$\mathbf{42.88 \pm 0.81\%}$	$\mathbf{18.29 \pm 1.23\%}$
0.0	256		$96.83 \pm 0.26\%$	$82.77 \pm 0.48\%$	$39.66 \pm 1.08\%$	$17.68 \pm 0.83\%$
0.9	100		$81.07 \pm 0.26\%$	$36.73 \pm 0.77\%$	$15.60 \pm 0.34\%$	$7.29 \pm 0.13\%$
0.9	256		$94.59 \pm 1.78\%$	$77.61 \pm 3.82\%$	$25.42 \pm 2.02\%$	$13.63 \pm 0.45\%$
0.0	100	200eps	$82.11 \pm 0.48\%$	$70.46 \pm 0.94\%$	$16.66 \pm 0.20\%$	$7.42 \pm 0.14\%$
0.0	256		$\mathbf{97.25 \pm 0.15\%}$	$\mathbf{85.21 \pm 0.56\%}$	$\mathbf{40.57 \pm 1.07\%}$	$\mathbf{18.13 \pm 0.33\%}$
0.9	100		$77.03 \pm 0.31\%$	$61.00 \pm 1.81\%$	$12.73 \pm 0.82\%$	$5.49 \pm 0.21\%$
0.9	256		$77.30 \pm 1.60\%$	$60.74 \pm 5.21\%$	$13.35 \pm 0.74\%$	$6.28 \pm 0.92\%$
Permuted Ticket			$81.66 \pm 0.68\%$	$70.98 \pm 1.08\%$	$15.63 \pm 0.13\%$	$6.29 \pm 0.30\%$
Trained Dense Network			$71.17 \pm 0.81\%$	$56.42 \pm 3.54\%$	$19.15 \pm 0.22\%$	$4.41 \pm 0.25\%$

Table 27: Transferability to different datasets for frozen 89.26% **ResNet-18** tickets extracted on **CIFAR-100** with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	MNIST	EuroSAT	CIFAR-10	TinyImageNet
0.0	100	200eps	84.47 ± 0.80%	73.05 ± 1.00%	38.17 ± 0.23%	8.74 ± 0.31%
0.0	256	200eps	**97.18 ± 0.20%**	**86.59 ± 0.54%**	**67.60 ± 0.89%**	**20.59 ± 0.97%**
0.9	100	200eps	79.71 ± 0.79%	66.08 ± 1.99%	34.95 ± 1.95%	6.78 ± 0.07%
0.9	256	200eps	79.34 ± 3.08%	68.76 ± 0.70%	35.84 ± 0.97%	7.29 ± 0.61%
Permuted Ticket			81.35 ± 1.75%	69.66 ± 0.92%	35.87 ± 0.70%	5.96 ± 0.12%
Trained Dense Network			90.10 ± 0.36%	67.04 ± 3.70%	75.39 ± 0.41%	13.16 ± 0.46%

Table 28: Transferability to different datasets for frozen 89.26% **VGG16** tickets extracted on **CIFAR-100** with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	MNIST	EuroSAT	CIFAR-10	TinyImageNet
0.0	100	20eps	**71.50 ± 1.83%**	**60.61 ± 1.82%**	**40.97 ± 0.73%**	**5.08 ± 0.47%**
0.0	256	20eps	65.03 ± 1.67%	54.02 ± 1.56%	38.55 ± 0.70%	4.72 ± 0.30%
0.9	100	20eps	24.18 ± 2.89%	20.47 ± 2.91%	15.62 ± 2.80%	0.70 ± 0.15%
0.9	256	20eps	64.83 ± 2.38%	51.25 ± 5.11%	37.84 ± 0.64%	3.76 ± 0.14%
0.0	100	200eps	56.43 ± 4.11%	52.81 ± 2.62%	22.84 ± 1.72%	3.11 ± 0.15%
0.0	256	200eps	**67.31 ± 1.32%**	**56.78 ± 1.60%**	**35.59 ± 0.51%**	**4.25 ± 0.43%**
0.9	100	200eps	13.35 ± 2.61%	31.61 ± 0.95%	12.45 ± 0.87%	1.63 ± 0.27%
0.9	256	200eps	14.52 ± 1.99%	34.39 ± 3.60%	14.52 ± 1.99%	1.51 ± 0.19%
Permuted Ticket			11.80 ± 0.09%	31.89 ± 4.04%	12.03 ± 0.23%	1.29 ± 0.13%
Trained Dense Network			63.22 ± 0.96%	48.74 ± 3.69%	57.18 ± 0.42%	5.08 ± 0.41%

Table 29: Transferability to different datasets for frozen 89.26% **Swin** tickets extracted on **TinyImageNet** with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	EuroSAT	CIFAR-10	CIFAR-100	CUB-200
0.0	100	200eps	56.83 ± 0.46%	42.54 ± 0.23%	16.48 ± 0.46%	3.69 ± 0.15%
0.0	256	200eps	**62.57 ± 1.43%**	**44.61 ± 0.37%**	**18.68 ± 0.46%**	**4.42 ± 0.33%**
0.9	100	200eps	55.57 ± 0.84%	41.48 ± 0.76%	16.49 ± 1.70%	3.37 ± 0.15%
0.9	256	200eps	55.82 ± 2.81%	41.75 ± 1.53%	16.62 ± 1.88%	3.26 ± 0.19%
Permuted Ticket			54.70 ± 0.26%	41.55 ± 0.24%	17.69 ± 0.29%	6.97 ± 0.11%
Trained Dense Network			91.81 ± 0.27%	74.95 ± 0.41%	52.84 ± 0.12%	52.57 ± 0.23%

Table 30: Transferability to different datasets for frozen 89.26% ResNet-34 tickets extracted on TinyImageNet with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	EuroSAT	CIFAR-10	CIFAR-100	CUB-200
0.0	100	20eps	$92.04 \pm 0.42\%$	**68.11 ± 0.69%**	$45.63 \pm 0.76\%$	$14.08 \pm 0.17\%$
0.0	256	20eps	**92.37 ± 0.26%**	$67.84 \pm 0.59\%$	**45.95 ± 0.56%**	**14.78 ± 0.39%**
0.9	100	20eps	$65.75 \pm 0.63\%$	$30.21 \pm 0.52\%$	$45.63 \pm 0.76\%$	$3.21 \pm 0.23\%$
0.9	256	20eps	$73.45 \pm 0.74\%$	$40.17 \pm 1.25\%$	$17.53 \pm 0.50\%$	$4.14 \pm 0.14\%$
0.0	100	200eps	$63.25 \pm 0.76\%$	$27.87 \pm 0.60\%$	$10.25 \pm 0.63\%$	$3.57 \pm 0.27\%$
0.0	256	200eps	**78.19 ± 0.30%**	**46.72 ± 1.98%**	**22.86 ± 1.06%**	**5.74 ± 0.30%**
0.9	100	200eps	$46.19 \pm 8.49\%$	$26.54 \pm 1.83\%$	$6.98 \pm 1.12\%$	$3.24 \pm 0.16\%$
0.9	256	200eps	$54.25 \pm 5.52\%$	$26.83 \pm 2.69\%$	$8.87 \pm 1.71\%$	$3.92 \pm 1.34\%$
Permuted Ticket			$65.03 \pm 4.09\%$	$30.72 \pm 0.60\%$	$12.13 \pm 0.72\%$	$3.89 \pm 0.35\%$
Trained Dense Network			$90.28 \pm 0.52\%$	$65.11 \pm 0.19\%$	$44.24 \pm 0.02\%$	$20.92\% \pm 0.57\%$

Table 31: Transferability to different datasets for frozen 89.26% **ResNet-50** tickets extracted on **ImageNet100** with different configurations.

Hyperparameters			Target dataset			
μ	b	Budget	CIFAR-10	CIFAR-100	TinyImageNet	CUB-200
0.0	100	200eps	$37.36 \pm 1.08\%$	$15.31 \pm 0.04\%$	$10.37 \pm 0.47\%$	$8.25 \pm 0.25\%$
0.0	256	200eps	**45.35 ± 0.76%**	**21.96 ± 0.94%**	**18.26 ± 1.03%**	**13.36 ± 0.76%**
0.9	100	200eps	$20.45 \pm 1.35\%$	$4.98 \pm 0.68\%$	$2.38 \pm 0.27\%$	$2.78 \pm 0.55\%$
0.9	256	200eps	$21.55 \pm 0.72\%$	$5.70 \pm 0.48\%$	$2.65 \pm 0.33\%$	$2.85 \pm 0.30\%$
Permuted Ticket			$23.69 \pm 1.30\%$	$6.78 \pm 0.77\%$	$2.89 \pm 0.23\%$	$3.11 \pm 0.25\%$
Trained Dense Network			$53.88 \pm 1.34\%$	$31.65 \pm 1.00\%$	$33.57 \pm 0.30\%$	$33.88 \pm 0.24\%$

F.3 Few-shot Generalization

By limiting the number of data samples in the *Sparse Training* phase, we can see how well tickets found with different hyperparameter settings generalize when less data is available. In our experiments we subsample the dataset via the random selection method – as this has been shown to work best for small subsets (Guo et al., 2022) while being also computationally efficient to calculate – using the following rate $[1\%, 2\%, 5\%, 10\%]$, while making sure that the resulting subset is class-balanced. We additionally compare with two baselines, namely a fully dense network, and a network obtained by layerwise permutation of the sparsity mask of a Lottery Ticket, thus having the same layerwise sparsity and initialization. We show results for the 89.26% sparsity in Tables 32 to 37.

Closer tickets generalize better with limited data. We observe that closer tickets can recover a significantly higher proportion of the full dataset accuracy when trained on a dataset subset. More specifically for ResNet-34 on TinyImageNet, they can recover $\sim 65\%$ of the full dataset accuracy given 10 times less data, where other networks recover less than 50%. Limiting the training budget of the *Mask Search* phase to 20eps results in an even more impressive recovery of $\sim 78\%$, while having a negligible impact for the other hyperparameters.

Table 32: Validation accuracies of a 89.26% sparse **ResNet-18** ticket when trained on a **CIFAR-10** subset.

Hyperparameters			Subset sizes			
μ	b	Budget	1%	2%	5%	10%
0.0	100	20eps	**87.75 ± 0.49%**	**88.79 ± 0.39%**	**89.95 ± 0.40%**	**90.84 ± 0.15%**
0.0	256		84.19 ± 0.55%	86.51 ± 0.17%	88.74 ± 0.05%	89.48 ± 0.15%
0.9	100		39.24 ± 1.51%	48.48 ± 0.98%	65.06 ± 0.97%	78.93 ± 1.47%
0.9	256		55.34 ± 5.67%	63.87 ± 6.16%	77.71 ± 3.32%	83.87 ± 1.20%
0.0	100	200eps	44.22 ± 1.47%	53.37 ± 1.41%	68.40 ± 0.96%	78.90 ± 0.30%
0.0	256		**85.43 ± 0.58%**	**88.03 ± 0.62%**	**90.58 ± 0.21%**	**91.82 ± 0.11%**
0.9	100		42.93 ± 2.12%	52.80 ± 3.05%	71.35 ± 1.49%	80.96 ± 0.29%
0.9	256		37.02 ± 1.03%	47.38 ± 1.35%	63.93 ± 4.25%	75.44 ± 1.05%
Permuted Ticket			40.94 ± 1.54%	50.25 ± 2.77%	67.28 ± 1.05%	80.03 ± 0.75%
Dense Network			42.54 ± 1.15%	51.25 ± 0.78%	67.15 ± 0.34%	78.42 ± 0.53%

Table 33: Validation accuracies of a 89.26% sparse **ResNet-18** ticket when trained on a **CIFAR-100** subset.

Hyperparameters			Subset sizes			
μ	b	Budget	1%	2%	5%	10%
0.0	100	200eps	11.20 ± 0.13%	16.71 ± 0.40%	28.61 ± 0.40%	42.01 ± 0.21%
0.0	256		**49.60 ± 0.53%**	**56.13 ± 0.84%**	**63.70 ± 0.26%**	**67.36 ± 0.44%**
0.9	100		8.63 ± 0.31%	14.23 ± 0.42%	24.13 ± 0.28%	37.31 ± 0.78%
0.9	256		9.24 ± 0.68%	13.43 ± 0.61%	22.54 ± 1.03%	35.58 ± 0.49%
Permuted Ticket			9.43 ± 0.33%	13.89 ± 0.25%	24.70 ± 1.22%	35.37 ± 4.50%
Dense Network			9.44 ± 0.48%	13.69 ± 0.45%	22.73 ± 0.09%	34.34 ± 0.48%

Table 34: Validation accuracies of a 89.26% sparse **VGG16** ticket when trained on a **CIFAR-100** subset.

Hyperparameters			Subset sizes			
μ	b	Budget	1%	2%	5%	10%
0.0	100	20eps	**43.06 ± 0.95%**	**48.57 ± 0.59%**	**54.17 ± 0.21%**	**58.66 ± 0.19%**
0.0	256		37.13 ± 0.78%	43.17 ± 0.63%	49.20 ± 0.39%	52.56 ± 0.38%
0.9	100		9.12 ± 0.67%	13.30 ± 0.38%	26.47 ± 1.59%	41.37 ± 0.49%
0.9	256		22.04 ± 1.09%	33.37 ± 0.81%	47.52 ± 0.78%	54.95 ± 0.64%
0.0	100	200eps	12.07 ± 0.54%	16.80 ± 0.37%	27.60 ± 0.63%	37.82 ± 0.77%
0.0	256		**36.43 ± 1.11%**	**44.73 ± 0.07%**	**54.51 ± 0.15%**	**59.64 ± 0.19%**
0.9	100		7.67 ± 0.26%	12.17 ± 0.18%	25.03 ± 1.19%	40.26 ± 0.76%
0.9	256		7.91 ± 0.35%	12.65 ± 0.83%	23.81 ± 0.78%	38.84 ± 1.14%
Permuted Ticket			9.93 ± 0.27%	14.01 ± 0.51%	27.24 ± 0.80%	42.07 ± 0.77%
Dense Network			5.25 ± 0.64%	9.27 ± 0.36%	20.59 ± 1.01%	39.26 ± 0.67%

Table 35: Validation accuracies of 89.26% sparse ResNet-34 tickets when trained on a TinyImageNet subset.

Hyperparameters			Subsampling rate			
μ	b	Budget	1%	2%	5%	10%
0.0	100	20eps	$20.70 \pm 0.41\%$	$27.06 \pm 0.40\%$	$37.09 \pm 0.20\%$	$44.13 \pm 0.54\%$
0.0	256	20eps	$\mathbf{28.49 \pm 0.53\%}$	$\mathbf{33.06 \pm 0.68\%}$	$\mathbf{39.69 \pm 0.20\%}$	$\mathbf{46.18 \pm 0.54\%}$
0.9	100	20eps	$3.89 \pm 0.19\%$	$6.77 \pm 0.25\%$	$13.56 \pm 0.68\%$	$24.74 \pm 1.09\%$
0.9	256	20eps	$4.75 \pm 0.38\%$	$8.02 \pm 0.63\%$	$16.20 \pm 0.66\%$	$24.81 \pm 1.15\%$
0.0	100	200eps	$4.72 \pm 0.46\%$	$7.69 \pm 0.39\%$	$16.46 \pm 0.24\%$	$27.34 \pm 0.18\%$
0.0	256	200eps	$\mathbf{10.49 \pm 1.14\%}$	$\mathbf{17.90 \pm 1.72\%}$	$\mathbf{29.51 \pm 1.71\%}$	$\mathbf{37.87 \pm 0.43\%}$
0.9	100	200eps	$3.41 \pm 0.28\%$	$5.18 \pm 0.59\%$	$14.33 \pm 0.55\%$	$8.36 \pm 13.61\%$†
0.9	256	200eps	$3.40 \pm 0.62\%$	$6.17 \pm 0.85\%$	$11.42 \pm 0.31\%$	$21.25 \pm 1.63\%$
Permuted Ticket			$4.34 \pm 0.20\%$	$6.33 \pm 0.48\%$	$13.75 \pm 0.43\%$	$26.30 \pm 0.31\%$
Dense Network			$4.32 \pm 0.13\%$	$6.33 \pm 0.19\%$	$13.19 \pm 0.33\%$	$25.13 \pm 0.57\%$

Table 36: Validation accuracies of a 89.26% sparse **Swin** ticket when trained on a **TinyImageNet** subset.

Hyperparameters			Subset sizes			
μ	b	Budget	1%	2%	5%	10%
0.0	100	200eps	$6.01 \pm 0.29\%$	$11.07 \pm 0.13\%$	$18.20 \pm 0.32\%$	$24.25 \pm 0.66\%$
0.0	256	200eps	$\mathbf{7.28 \pm 0.17\%}$	$\mathbf{12.52 \pm 0.49\%}$	$\mathbf{21.88 \pm 0.47\%}$	$\mathbf{29.95 \pm 0.19\%}$
0.9	100	200eps	$5.40 \pm 0.42\%$	$8.47 \pm 0.24\%$	$14.69 \pm 0.56\%$	$20.18 \pm 0.70\%$
0.9	256	200eps	$5.89 \pm 0.18\%$	$9.03 \pm 0.48\%$	$14.65 \pm 0.45\%$	$20.74 \pm 0.47\%$
Permuted Ticket			$6.01 \pm 0.23\%$	$9.13 \pm 0.20\%$	$14.94 \pm 0.58\%$	$20.13 \pm 0.26\%$
Dense Network			$5.47 \pm 0.48\%$	$7.34 \pm 0.68\%$	$13.36 \pm 0.46\%$	$19.51 \pm 0.60\%$

Table 37: Validation accuracies of a 89.26% sparse **ResNet-50** ticket when trained on a **ImageNet100** subset.

Hyperparameters			Subset sizes			
μ	b	Budget	1%	2%	5%	10%
0.0	100	200eps	$37.89 \pm 2.84\%$	$51.15 \pm 2.50\%$	$65.67 \pm 0.33\%$	$72.75 \pm 0.30\%$
0.0	256	200eps	$\mathbf{43.80 \pm 3.10\%}$	$\mathbf{58.81 \pm 0.88\%}$	$\mathbf{67.06 \pm 0.80\%}$	$\mathbf{72.76 \pm 0.43\%}$
0.9	100	200eps	$9.76 \pm 1.32\%$	$16.45 \pm 1.41\%$	$30.86 \pm 3.50\%$	$46.00 \pm 1.71\%$
0.9	256	200eps	$4.13 \pm 2.06\%$	$15.47 \pm 1.43\%$	$30.41 \pm 0.82\%$	$45.49 \pm 0.12\%$
Permuted Ticket			$10.69 \pm 1.51\%$	$7.03 \pm 9.30\%$	$11.86 \pm 16.28\%$	$18.37 \pm 30.09\%$
Dense Network			$4.28 \pm 0.58\%$	$7.88 \pm 0.48\%$	$26.77 \pm 3.64\%$	$52.36 \pm 2.82\%$

G HYPERPARAMETER CONFIGURATIONS

General hyperparameters. Each training run contains of 200 (90 for ImageNet100) epochs with a chosen batch size and momentum parameter. Throughout training the learning rate starts at 0.1 (0.2 for TinyImageNet) and is cosine annealed. We also apply a weight decay of 1E-4. Each dataset uses normalization, random cropping and horizontal flipping as data augmentation.

LTH hyperparameters. Each iteration 20% of the unpruned parameters are pruned. Late-rewinding involves training for 2 epochs to determine the rewinding point. Each experiments consists of 25 (20 for TinyImageNet, 15 for ImageNet-100) rounds of IMP.

Linear Probing. Each linear probe is trained for 5 epochs with a learning rate of 0.005 and weight decay of 1E-4. The probes are trained with $\mu = 0.9$ and $b = 256$. No LR scheduling is used.