# The Influence of SGD noise on Lottery Ticket performance

**Anonymous authors**
**Paper under double-blind review**

## Abstract

The Lottery Ticket Hypothesis states that there exist sparse subnetworks (called 'winning' Lottery Tickets) within dense 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, an additional pretraining step is required for winning lottery tickets to be successfully found. Previous work linked the amount of pretraining required to a measurement of instability to SGD noise. In this paper, we take a closer look at the training hyperparameters that influence SGD instability during normal training and link this to the ability to find 'winning' tickets. We show that several techniques that have a positive influence on dense network generalization increase SGD instability, and as such hinder the extraction of 'winning' tickets. By dampening this instability via smart hyperparameter selection, we show that we can extract 'winning' tickets without pretraining and even outperform tickets found with pretraining at more extreme sparsities. We furthermore discover that tickets found with less instability to SGD noise have as unexpected side effect that they encode powerful features for classification in the untrained weights. We show that these features do not emerge when extracted under more unstable hyperparameter settings, and that they are transferable to different datasets, as well as enable faster training of the resulting tickets.

## 1 Introduction

In a recent study by Sevilla et al. (2022) it was calculated that the computational resources to train a state-of-the-art neural network model have doubled roughly every 4 to 9 months. Since then, Large Language Models have become more ubiquitous, leading to the reality that these models can only be trained by the largest research centra and corporations, as they require enormous amounts of computation to train.

There exist however techniques to reduce the cost of training a model. These focus either on intelligently selecting the training data for the model with techniques such as coreset selection (Guo et al., 2022) or dataset distillation (Wang et al., 2018; Zhao et al., 2021), or reducing the number of trainable parameters. This reduction can be achieved in a number of ways, such as using a larger model to guide a smaller model via Knowledge Distillation (Hinton et al., 2015), training multiple small models in parallel (Zhang et al., 2018), weight quantization (Vanhoucke et al., 2011), or generating sparse networks (Han et al., 2015; Li et al., 2017; Frankle & Carbin, 2019; Lee et al., 2019; Ramanujan et al., 2020).

A promising approach to generate sparse networks is Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2019). This hypothesis states that within a randomly initialized dense network, there exist a sparse network that can be trained to similar accuracy as the dense network under the same training regime. While the commonly-used approach to find these networks (called 'Winning Tickets') is an intensive iterative procedure called Iterative Magnitude Pruning (IMP), the sparsity of the resulting networks cannot be matched by other approaches without compromising the final accuracy. As such, techniques that allow for speeding up IMP, or increasing our understanding of the functionality behind it are highly valuable.

It was noted in (Frankle et al., 2020) that Lottery Tickets for more complex dataset-network settings could not be found directly in random initializations. Rather, they discovered that in such cases, the IMP needed to be applied on a lightly pretrained model, which they term late rewinding, or Lottery Ticket Rewinding

(LTR). The authors link this phenomenon to instability to SGD noise, and quantify this instability by measuring the maximum error on the interpolation path between two solutions found via SGD starting from the same initialization. The application of this pretraining step has been treated as a given, with limited research on other approaches to limit SGD noise and how it impacts the existence of winning lottery ticket.

### 1.1 Contributions

- By calculating the instability to SGD noise of a dense network for different hyperparameters settings, we show that certain hyperparameters limit SGD instability, but that this is linked to a reduced generalization error.
- By considering these hyperparameters in IMP, we show that less instability leads to less accuracy degradation when pruning, and as such to tickets that perform better at high sparsities, or even 'winning' tickets without pretraining.
- With careful hyperparameter selections, we can find Lottery Tickets that outperform tickets found with late rewinding under other hyperparameter configurations, albeit at extreme sparsities.
- We discover that tickets found with increased SGD stability can function as feature extractors when frozen, showing that IMP is a form of feature selection. These features additionally show a remarkable capacity for generalization to other datasets.
- Building upon the previous observation, we notice that these tickets can be trained to a surprisingly competitive accuracy with only a small subset of the dataset.

## 2 Related Work

### 2.1 The Lottery Ticket Hypothesis

**Analysis and extensions.** Since the introduction of the Lottery Ticket Hypothesis (LTH), and the Iterative Magnitude Pruning (IMP) algorithm by Frankle & Carbin (2019), the phases that make up IMP have been analyzed and extended. Starting with Zhou et al. (2019) which listed three distinct components in the *Mask Search* phase, namely the Mask Criterion, the Mask-1 action, and the Mask-0 action, and performed an ablation study across these components. Frankle et al. (2020) discovered that Lottery Tickets were unable to be found in more complex settings. This was remedied by introducing late rewinding, applying IMP on a lightly pretrained network. Both (Zullich et al., 2021; You et al., 2020) introduce methodologies to speed the Mask Search phase. In the case of (Zullich et al., 2021), this is with the introduction of Accelerated Iterative Magnitude Pruning (AIMP), while (You et al., 2020) introduces Early-bird tickets. Paul et al. (2022) introduces a loss landscape geometry interpretation of IMP and used this to answer several questions related to the working of IMP.

Our analysis is also centered on the *Mask Search* phase, and more specifically the parameters which influence the instability of 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 be still be found where previously late rewinding was required.

**Lottery Tickets Features.** A number of studies has 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 tested 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.

In this paper, we go a step further and measure the expressivity of the untrained ticket by finetuning a linear layer directly on the target dataset.

### 2.2 (Linear) Mode Connectivity.

Garipov et al. (2018); Draxler et al. (2018) simultaneously discovered that neural network solutions can be connected in the loss landscape via a continuous path such that the error across the path does not exceed a certain threshold. In (Garipov et al., 2018), this has been shown to be valid for polynomial paths with a single bend and quadratic Bezier curves. Linear Mode Connectivity goes a step further and requires the path to be linear. Generally, this property has been observed in two distinct cases. The first involves two 'child' models that continue training with different SGD noise from a slightly trained initialization. This paradigm has been introduced in (Frankle et al., 2020). Fort et al. (2020) uses the Neural Tangent Kernel to show that early in training the chaotic behavior of SGD determines a loss basin, in which later iterations optimize, thus providing empirical evidence for the emergence of linear mode connectivity. The second approach introduced in (Entezari et al., 2022) posits that with intelligent weight permutation, all solutions found by SGD lie in the same loss basin, i.e., are linear connected without error barrier. Ferbach et al. (2024) uses optimal transport to prove this with some added restrictions on the weight distribution.

In this research, we study the impact of different hyperparameters on Linear Mode Connectivity, by quantifying instability as defined in Frankle et al. (2020).

## 3 Methodology

We start from commonly used training configurations (see appendix) for the Lottery Ticket Hypothesis, taken from Frankle et al. (2021), and study the impact of different hyperparameters on train-time metrics, such as validation accuracy, Linear Mode Connectivity (Frankle et al., 2020), forgetting scores (Toneva et al., 2019), and convergence speed. In the following paragraphs we will first highlight the specific hyperparameters we study, and next we list the train-time metrics we measure.

### 3.1 The Lottery Ticket Hypothesis

**Algorithm 1** Iterative Magnitude Pruning with late rewinding.

1: Initialize a neural network with weights $\theta_0 \in \mathbb{R}^d$.
2: Initialize pruning mask $M = 1^d$.
3: Train $\theta_0$ for $p$ steps to $\theta_p$. ▷ *Pretraining*
4: **for** $n \in \{1, \ldots, N\}$ **do** ▷ *Mask Search*
5: Train $M \odot \theta_p$ to convergence.
6: Prune the k% unpruned weights with lowest magnitude.
Let $M[i]$=0 if the corresponding weight $i$ is pruned.
7: **end for**
8: Train the final network $M \odot \theta_p$. ▷ *Sparse Training*

Defined by an iterative approach (see Algorithm 1), the procedure to generate Lottery Tickets consists of two (or three) phases. In the optional *Pretraining* phase, the network with weights $\theta_0$ is lightly trained to provide a more stable rewind point (Frankle et al., 2020) for the next phase, resulting in weights $\theta_p$. The second phase is a *Mask Search* phase, in which the sparse network is trained to convergence with the same training configuration as the dense network. The goal of this training is to identify parameters for pruning by determining the lowest magnitude parameters at the end of this phase. Next, a fixed percentage (k) of the lowest magnitude parameters are pruned by setting the pruning mask $M$ for those weights to 0, and the remaining weights are reset to the rewind point (either the initialization $\theta_0$ or the pretrained weights $\theta_p$). These steps are repeated N times. Finally, in the *Sparse Training* phase the ticket is trained until completion, after which the network is usable in prediction. A winning lottery ticket is then defined as a pruned network that can attain similar (or better) validation accuracy as the dense network in commensurate training.

It has been shown in Frankle et al. (2021); Vischer et al. (2022) that the trainability of a ticket is dependent on both the initialization and the specific pruning mask. This directly translates to a dependency on the *Pretraining* phase for the initial weights and a dependency on the *Mask Search* phase for the pruning

mask. In this research we specifically focus on the hyperparameters used in the *Mask Search* phase. As a comprehensive grid search on all possible parameters is prohibitively expensive due to the costly, iterative nature of this procedure, we instead focus our attention on a few parameters and their impact. This will allow us to better understand the impact on the trainability and predictive quality of the resulting lottery tickets.

**Batch Size.** In the case of large datasets, it is often infeasible to use the whole dataset during each update step of the network. As such, it is common to use minibatches of the dataset to update a network iteratively via Stochastic Gradient Descent. Intuitively, if the batch size is smaller, then the resulting gradient is more influenced by the subsampling of the dataset, as the batch composition can be dramatically different from the composition of the full dataset. However, it has been shown in (Keskar et al., 2017) that smaller batch sizes positively impact the generalization of a neural network on unseen data.

**Momentum.** Introduced as a technique to speed up gradient descent, momentum allows gradient information to be carried over from previous batches in the weight update by employing a factor $\mu \in [0, 1]$ to weight the gradient information of the previous batch. As such, the updates function similar to a exponential moving average (see Equations (1) and (2)). The application of momentum has been demonstrated to lead to better generalizing networks, and it has been posited by (Jelassi & Li, 2022) that this is due to the resulting classifiers generalizing on small-margin samples, rather than memorizing those samples. The authors additionally argue that the impact of momentum is more significant with higher batch sizes.

$$v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t) \tag{1}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{2}$$

**Training duration.** Training a network is a process with many sources of noise such as mini batch gradients, data augmentation and more. It is evident that this noise accumulates during repeated training epochs. As such, one method to limit SGD noise during training of a network is to simply limit the computational budget of the training phase, rather than tackling the sources of noise.

### 3.2 Train-time Metrics

**Instability to SGD noise.** Introduced by Frankle et al. (2020), this metric measures the relative increase in classification error on a linear interpolating path between two sets of network weights trained from the same initialization with different sets of SGD noise. An initialization is said to be unstable if there is a significant error barrier along the interpolation path. Given two networks $W_1$, $W_2$, with the validation error of a network measured as $\mathcal{E}(W)$, then the error barrier across an interpolating path between $W_1$ and $W_2$ for $\alpha \in [0, 1]$ is defined in Equation (3).

$$sup_\alpha \ \mathcal{E}(\alpha W_1 + (1 - \alpha) W_2) - \frac{\mathcal{E}(W_1) + \mathcal{E}(W_2)}{2} \tag{3}$$

In practice, as defined by (Frankle et al., 2020), a significant error barrier is seen as being higher than 2%. In this case, the network is seen as *unstable* to SGD noise, otherwise it is seen as *stable* to SGD noise.

**Forgetting events.** During training, a network learns to predict labels for its input samples. However, the learning process is not monotonic, meaning that if a sample is predicted correctly at some iteration $i$, it is possible that at iteration $i + k$ it is no longer correctly predicted. To measure these 'forgetting events', Toneva et al. (2019) introduced forgetting scores, which record whether a sample is learned correctly each time the network is fed the sample. Samples that are never forgotten once learned are called 'unforgettable samples'.

**Convergence speed.** By calculating the area under the error curve during a training run of $T$ epochs, we can measure the convergence speed of a training configuration. Specifically, we calculate the error $\mathcal{E}(W^0)$ before training and after every epoch ($\mathcal{E}(W^{ep})$). We then average $\mathcal{E}(W^{ep}) - \mathcal{E}(W^0)$ over the duration of the training. To account for different optimal accuracies, we rescale this value by dividing with $\mathcal{E}(W^T) - \mathcal{E}(W^0)$.

$$\frac{1}{T}\sum_{ep=1}^{T}\frac{\mathcal{E}(W^0)-\mathcal{E}(W^{ep})}{\mathcal{E}(W^0)-\mathcal{E}(W^T)} \tag{4}$$

### 3.3 Experimental setup

For the experiments, we focus on ResNet models (He et al., 2016), both ResNet-18 and ResNet-34, which are trained on CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015). We resort to these datasets and networks, as they are commonly used within the literature. For the transferability experiments in Section 5.1, we additionally use MNIST LeCun et al. (2010) and EuroSAT Helber et al. (2019). Most experiments in the main paper list the results for ResNet-34 + TinyImageNet unless otherwise indicated, as this is the most complex setting, but we list results for the other settings in the appendix. Unless otherwise mentioned, each experiment is repeated for three runs.

We use configurations from T et al. (2022) as a starting point. More specifically, this means that we use a total training budget of 200 epochs in each iteration, after which we prune 20% of the weights. In the case of late Rewinding, this budget includes a *Pretraining* phase of 2 epochs. Training is done by minimizing the Cross-Entropy Loss with SGD, starting with a learning rate of 0.1 (or 0.2 for TinyImageNet) which is cosine annealed, and a weight decay of 5E-4. A full breakdown of the training configuration can be found in the appendix.

## 4 Experiments

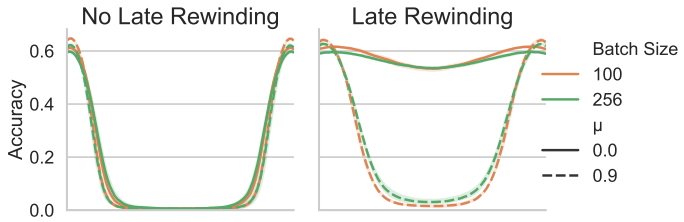
### 4.1 Instability in Dense Networks

We explore the influence of the momentum and batch size parameters on train-time performance of ResNet-34 models on TinyImageNet. For this, we start with a randomly initialized network and use different sets of SGD noise to achieve three versions of a trained network. In Figure 1 we show the average pairwise interpolation curves and error barriers for the three possible combinations. The results show that when starting from the random initialization (without late rewinding), no configuration can be deemed stable to SGD noise. When starting from a set of pretrained weights, the error barrier decreases significantly for the approaches without momentum ($\mu = 0.9$, solid lines), but only marginally decreases for those with momentum ($\mu = 0.9$, dotted lines). Additionally, the approaches that experience less instability, also have a (slightly) reduced generalization (see 0.00% sparsity column in Table 2). This hints at a relation between instability during the training of dense neural networks and the final performance. In the appendix we further explore the amount of pretraining required for each set of hyperparameters to achieve stability, where we show that settings with lower batch size, or momentum require more pretraining to be stable.

**Example Forgetting.** When observing the results of the example forgetting metric (Table 1), we can see a link between an increase in forgetting events on one side and a decrease in unforgettable samples, and an increase in SGD instability on the other side. In general, the application of momentum seems to have a much higher impact on the number of forgetting events than a different batch size has. We hypothesize that the frequent forgetting and re-learning of certain samples might steer the model to better generalization. As observed in (Jelassi & Li, 2022), momentum can lead models to generalize rather than memorize certain difficult samples, which causes better validation accuracy.

In the appendix, we dig deeper in the relationship between example forgetting and SGD instability by showing that removing frequently-forgotten samples decreases instability at the cost of validation accuracy.

**Convergence speed.** We can see that the approaches without momentum have a faster convergence than the approaches with momentum. This further highlights that while training with momentum results in a higher final accuracy, this is due to a more exploratory learning approach, as it takes significantly more training epochs for the momentum-based approaches to attain close to their final accuracy.




(a)

| | | Error Barrier | |
|---|---|---|---|
| $\mu$ | Batch | w/o late rewinding | w/ late rewinding |
| 0.0 | 100 | 60.26 ± 0.10 | 7.50 ± 0.65 |
| 0.0 | 256 | 58.98 ± 0.05 | 5.42 ± 0.05 |
| 0.9 | 100 | 63.56 ± 0.05 | 61.85 ± 0.22 |
| 0.9 | 256 | 60.92 ± 0.30 | 58.84 ± 0.86 |

(b)

Figure 1: Instability for a dense ResNet-34 model trained on TinyImageNet with different training configurations. We show the full interpolation curve in **(a)** and the error barrier in **(b)**.

Table 1: Average forgetting events per sample, proportion of unforgettable samples and convergence statistics for different hyperparameter configurations used in training ResNet-34 models on TinyImageNet.

| $\mu$ | Batch Size | Forgetting events | Unforgettable % | Convergence |
|---|---|---|---|---|
| 0.0 | 100 | 34.73 ± 12.54 | 0.23 ± 0.02% | 0.907 ± 0.003 |
| 0.0 | 256 | 32.83 ± 11.90 | 0.27 ± 0.02% | 0.771 ± 0.109 |
| 0.9 | 100 | 42.78 ± 14.37 | 0.05 ± 0.00% | 0.773 ± 0.042 |
| 0.9 | 256 | 39.38 ± 13.19 | 0.11 ± 0.02% | 0.800 ± 0.111 |

### 4.2 Impact on Lottery Tickets

In Table 2, we have listed the validation accuracies of Lottery Tickets found under different hyperparameter configurations at different sparsities (after every 5 pruning steps). In the next paragraphs, we use these results to highlight how parameter selection and the associated SGD instability of the training procedure impacts the resulting Lottery Tickets.

**Instability in relation to ticket performance.** Even though all approaches without late rewinding roughly have the same instability as determined by the error barrier, we can see significant differences in how the ticket performances evolve at different sparsity levels. At lower sparsities, we see that approaches with lower batch size or $\mu = 0.9$ have higher validation accuracy which deteriorates faster at higher sparsities. Approaches with $\mu = 0.0$ or with higher batch sizes initially have lower validation accuracy, but those approaches suffer less accuracy drop, or even increase in accuracy when pruning. Interestingly, some settings feature 'winning' tickets, which is unexpected as stability to SGD noise has been introduced specifically to find those, but none of these settings are deemed stable to SGD noise.

These observations, lead us to believe that stability in the form of a low error barrier is not necessary for ticket performance. In fact, as there are configurations that result in the same error barrier, while having different performances, we believe that a lot of information is hidden behind this metric. We can see this in Figure 1b when we use late rewinding with two epochs of pretraining. Even though each hyperparameter configuration has roughly the same instability at initialization, the approaches with $\mu = 0.0$ have significantly less instability for the pretrained weights, while those with $\mu = 0.9$ experience only a minor decrease. This shows that instability as defined by the error barrier decreases slower for some configurations than others.

In the appendix, we show results for other dataset and network combinations, where we can make similar observations w.r.t. stability and ticket performance.

### 4.3 Limiting Mask Search budget

In each iteration of *Mask Search*, the network is trained for a number of epochs, during which SGD noise accumulates, before the network is pruned and rewound. As such, we could expect that limiting the number of epochs trained in this phase, can limit the instability incurred by these tickets. In fact, it has been observed by (Zullich et al., 2021) that it is possible to limit the number of epochs in the *Mask Search* phase without

Table 2: Lottery Ticket extracted from ResNet-34 on TinyImageNet performances for different training parameters at different sparsity levels. Starred(*) entries encompass one or multiple runs in which the ticket did not converge and remained at random chance.

| Ticket Parameters | | | Sparsity | | | | |
|---|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | 0.00% | 67.23% | 89.26% | 96.48% | 98.85% |
| 0.0 | 100 | ✗ | 61.14% | 59.76% | 57.79% | 54.47% | 50.98% |
| 0.0 | 256 | | 59.33% | 59.95% | **60.20%** | **58.89%** | **57.09%** |
| 0.9 | 100 | | **63.74%** | **62.46%** | 35.95%* | 35.69%* | 15.09%* |
| 0.9 | 256 | | 61.57% | 60.37% | 56.93% | 52.76% | 47.34% |
| 0.0 | 100 | ✓ | 60.91% | 62.90% | **62.67%** | **60.37%** | 56.11% |
| 0.0 | 256 | | 59.27% | 60.01% | 60.08% | 58.59% | **56.40%** |
| 0.9 | 100 | | **63.45%** | **63.02%** | 60.10% | 57.73% | 35.10%* |
| 0.9 | 256 | | 61.66% | 61.22% | 58.95% | 55.91% | 52.49% |

Table 3: AIMP results for different hyperparameter settings of ResNet-34 on TinyImageNet.

| Ticket Parameters | | | | Sparsity | | | |
|---|---|---|---|---|---|---|---|
| Budget | $\mu$ | Pretrained | Batch Size | 67.23% | 89.26% | 96.48% | 98.85% |
| 20eps | 0.0 | ✗ | 256 | 58.90% | 56.95% | 54.13% | 49.73% |
| | 0.9 | ✗ | 100 | **62.82%** | **61.41%** | **58.49%** | **53.65%** |
| 50eps | 0.0 | ✗ | 256 | 58.74% | 56.75% | 53.22% | **49.58%** |
| | 0.9 | ✗ | 100 | **63.02%** | **59.51%** | **54.04%** | 39.04% |

significant impact on the resulting ticket performance, which they term Accelerated Iterative Magnitude Pruning (AIMP). We study this finding on more complex settings, with computational budgets of 20 and 50 epochs in Table 3, which represents 10% and 25%, respectively, of the original budget. We additionally modify the LR schedule to reflect upon this limited budget.

Given a small training budget, we can find tickets via AIMP that perform similar, or outperform those found with IMP at higher sparsity levels. This observation only holds for the hyperparameter settings which we deemed more unstable, as we can see a noticeable decrease in accuracy compared to those found with IMP for the more stable settings. At higher compute budgets, the advantage of AIMP over IMP in unstable regimes decreases substantially, showing that indeed the SGD noise accumulation over the larger durations of the *Mask Search* phase is significant and has a negative effect at higher sparsities. In the more stable regimes, little performance difference is observed between the two budgets.

## 5 Consequences of stability in Lottery Tickets

### 5.1 Untrained tickets can encode useful features

Earlier work by Zhou et al. (2019) likened the Mask Search procedure to training a model. They determined that it is possible to generate a mask for a of set of network weights, such that the network achieves better than random chance predictions, effectively outperforming a randomly initialized network. Such masks are called supermasks, and have been extracted for simple convolutional models on CIFAR-10, MNIST. Further accuracy gains for these supermasks could be achieved by directly optimizing the masks via gradient descent, which can be likened to the Strong Lottery Ticket Hypothesis (Ramanujan et al., 2020) where a mask is optimized for a set of weights to approximate the performance a fully trained network.

The phenomenon of supermasks is quite restrictive, as it assumes that by pruning; *(i)* useful features emerge within the feature extractor, and *(ii)* the classification layer can exploit those features. All of this should

happen without modifying a single weight. In practice, during the earlier experiments we have not observed any ticket at meaningful sparsity that exhibited better than random chance without training.

Instead, we focus only on case *(i)*, meaning we still allow the training of a classification layer, but keep the pruned feature extractor frozen. Furthermore, we consider feature depth as well. Intuitively, if useful features are encoded within the ticket, then it should follow that the features in the deeper layers of the network are more expressive than those in the earlier layers, as is the case with a dense network. To determine this, we devise a linear evaluation experiment inspired by linear probing (Alain & Bengio, 2017).

**Linear feature evaluation.** After each ResNet block in the ticket we insert a linear probe, which is a channel-wise pooling operation followed by a linear classification layer. While the linear probe is training, we freeze all other parameters in the network, such that the resulting accuracy correctly reflects the predictive quality of the features. For consistency between different parameter configurations, we use a single set of hyperparameters to train the linear layer, as tests with different sets of hyperparameters only showed a minimal impact on the validation accuracy of the probe. We compare the results with a baseline, which consists of a ticket for which the mask is randomly layerwise permuted, and thus preserves the initialization and the layerwise sparsity.

In Table 4 we highlight the results at two sparsity levels, for two positions in the model (Blocks 4 & 8). The sparsity levels were chosen because of two reasons. First, they both correspond to the sparsity levels described in Table 2, and secondly at the 67.23% sparsity level, most tickets exhibit similar accuracy when fully trained. Block 4 is roughly equivalent to the middle of the model, while Block 8 considers the features at the last convolutional layer. We list the full results at all sparsities and blocks in the appendix.

First we highlight that the randomly permuted tickets do not significantly outperform the features encoded in a random dense network. We see a slight increase in accuracy at higher sparsity, but no major impact. This shows that sparsity by itself does not promote the emergence of useful features, and in fact too much 'random' sparsity can inhibit good features. Secondly, we can divide the features in tickets in three main categories. The majority of the results show no significant increase or decrease w.r.t. the dense network. That being said, tickets found without pretraining but with $\mu = 0.9$ typically have lower quality features encoded, while those found with $\mu = 0.0$ and pretraining have much higher quality features. We notice that the stable configurations result in potent feature extractors at initialization. This shows that the mask search procedure can find useful features from initialization or weights in early training. Interestingly, in the unstable cases the features at the end of the network provide no accuracy gain compared to the intermediate features, which is opposite to that of the more stable tickets (late rewinding with $\mu = 0.0$) where we can notice a clear increase, and what we would expect in a trained network. A simple comparison shows that the remarkable performance of stable tickets can not be explained by those factors alone.

**Transferability of emergent features.** Previous research Morcos et al. (2019); Mehta (2019) has demonstrated that Lottery Tickets generalize well to other datasets. We further explore our earlier observations that usable features emerge in lottery tickets, and analyze their transferability to different datasets. In this experiment, we choose to transfer tickets extracted for CIFAR-10 as this dataset is the least complex of the considered datasets. This implicates that the features encoded by the mask should be less suited for more complex datasets. To enable transferability to datasets with different number of classes, we replace the linear layer with a new linear layer with a target-specific number of outputs, and freeze all other layers. While this procedure leads to a loss of sparsity in the classification layer, this is the only possible approach short of resparsifying the linear layer, which might induce side effects.

Transferring is done to MNIST (LeCun et al., 2010), CIFAR-100 (Krizhevsky et al., 2009), TinyImageNet (Le & Yang, 2015) and EuroSAT (Helber et al., 2019). The reasoning for these datasets is as follows. MNIST is an easy dataset, albeit monochrome, which features different instances from CIFAR-10 (numbers, rather than objects). CIFAR-100 has been generated using the same process as CIFAR-10, but is more complex and features mutually exclusive images and classes. TinyImageNet is an even more challenging dataset, with more classes, less instances per class, and a higher resolution than CIFAR-10. Finally, EuroSAT features higher resolution landscape images, rather than object images. Each of these datasets has characteristics not present in CIFAR-10, which serves to demonstrate the versatility of the features present within the lottery

Table 4: Linear probing results at different locations in a ResNet-34 ticket extracted for TinyImageNet under different training configurations. Starred(*) entries correspond to dense features, and as such they have a sparsity of 0.00%.

| Ticket parameters | | | 67.23% sparsity | | 89.26% sparsity | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | Block 4 | Block 8 | Block 4 | Block 8 |
| 0.0 | 100 | ✗ | 4.26 ± 0.39% | 3.83 ± 0.19% | 3.97 ± 0.25% | 3.84 ± 0.27% |
| 0.0 | 256 | | **4.38 ± 0.35%** | **4.53 ± 0.47%** | **4.86 ± 0.26%** | **5.53 ± 0.70%** |
| 0.9 | 100 | | 1.96 ± 0.46% | 1.89 ± 0.27% | 1.29 ± 0.40% | 1.59 ± 0.40% |
| 0.9 | 256 | | 2.92 ± 0.48% | 2.63 ± 0.53% | 2.61 ± 1.20% | 2.04 ± 0.90% |
| 0.0 | 100 | ✓ | 6.66 ± 0.42% | 7.90 ± 1.38% | **6.39 ± 0.41%** | 7.24 ± 1.52% |
| 0.0 | 256 | | **6.67 ± 0.30%** | **9.55 ± 0.18%** | 6.32 ± 0.29% | **9.58 ± 0.42%** |
| 0.9 | 100 | | 4.55 ± 0.15% | 4.86 ± 0.46% | 4.23 ± 0.08% | 4.45 ± 0.30% |
| 0.9 | 256 | | 4.47 ± 0.21% | 4.53 ± 0.33% | 4.53 ± 0.12% | 4.73 ± 0.12% |
| Permuted ($\mu = 0.0$, BS=256) | | | 3.74 ± 0.12% | 3.39 ± 0.09% | 4.07 ± 0.09% | 4.09 ± 0.16% |
| Random dense network* | | | *3.66 ± 0.82%* | *3.13 ± 0.45%* | *3.66 ± 0.82%* | *3.13 ± 0.45%* |

ticket. During the network transfer, we normalize the images with the mean and standard deviation, and rescale to 32×32.

We show the results for 89.26% sparsity tickets in Table 5, where we additionally compare with several baselines. We finetune dense networks under several hyperparameter regimes to show the transferability of features specialized for a different dataset in unstable and less unstable regimes. We also provide an upper limit on the model performance by training a dense network from scratch on the target dataset.

We observe that the features learned in high instability regimes are significantly less transferable to other datasets, than those learned in more stable regimes. This is the case for both the dense network and also the derived Lottery Tickets, showing that this property is likely inherited by the ticket from the dense network. That being said, in no circumstances does transferring beat the retraining performance. Whether the ticket outperforms the dense network or not seems to be dependent on the dataset and instability. Stable tickets always outperform the dense network, but the most unstable tickets only do for certain datasets. In the appendix, we transfer CIFAR-100 and TinyImageNet tickets, and notice that while the validation accuracy of the transferred ticket is still remarkable, we fail to consistently outperform the features in the trained dense network. This is of course not that unexpected, seeing that 1) the tickets we transfer are still frozen at initialization and encode worse features than those of a dense network, as measured by our linear probes earlier, and 2) the features in the dense network are learned for a more complex setting and as such are already better transferable to less complex datasets.

### 5.2 Training tickets with limited data

Having observed this emergence of useful features to certain lottery tickets, we next want to determine how we can further exploit this. For this we consider accelerating the Sparse Training phase. Rather than using the full dataset, which can be costly and cannot be run on small devices, instead we consider few-shot learning setting, where a limited number of samples is used.

In our experiments, we use the following subset sizes: [1% , 2%, 5%, 10%]. As we focus on small dataset sizes, we will use the random selection method to select class-balanced subsets, since this has been shown to work best in those cases (Guo et al., 2022). To allow for comparability between TinyImageNet and CIFAR-10, we have chosen 1% as a bottom limit, as TinyImageNet features 500 images per class, while CIFAR-10 features 5000 images per class. Going lower than 1% could be feasible for CIFAR-10, but will be difficult for TinyImageNet, as then the randomness of the subset selection process will significantly impact

Table 5: Transferability of frozen ResNet-18 tickets extracted on CIFAR-10 with different configurations. Starred(*) entries correspond to dense trained networks, meaning the feature extractors have a sparsity of 0.00% and have been trained either on the source or the target dataset.

| Ticket parameters | | | Target dataset | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | MNIST | CIFAR-100 | TinyImageNet | EuroSAT |
| 0.0 | 100 | ✗ | 82.11 ± 0.48% | 16.66 ± 0.20% | 7.42 ± 0.14% | 70.46 ± 0.94% |
| 0.0 | 256 | | **97.25 ± 0.15%** | **40.57 ± 1.07%** | **18.13 ± 0.33%** | **85.21 ± 0.56%** |
| 0.9 | 100 | | 77.03 ± 0.31% | 12.73 ± 0.82% | 5.49 ± 0.21% | 61.00 ± 1.81% |
| 0.9 | 256 | | 77.30 ± 1.60% | 13.35 ± 0.74% | 6.28 ± 0.92% | 60.74 ± 5.21% |
| 0.0 | 100 | ✓ | 96.90 ± 0.03% | 36.18 ± 1.02% | 18.48 ± 0.79% | 82.40 ± 0.98% |
| 0.0 | 256 | | **97.45 ± 0.21%** | **41.63 ± 0.91%** | **21.56 ± 1.21%** | **85.47 ± 0.98%** |
| 0.9 | 100 | | 90.85 ± 1.60% | 17.34 ± 1.80% | 8.46 ± 0.29% | 69.37 ± 1.06% |
| 0.9 | 256 | | 92.79 ± 2.67% | 18.96 ± 4.22% | 10.47 ± 1.91% | 70.86 ± 1.43% |
| Frozen dense* ($\mu = 0.0$, BS=256) | | | *95.55 ± 0.15%* | *39.82 ± 0.12%* | *12.46 ± 0.20%* | *78.59 ± 0.81%* |
| Frozen dense* ($\mu = 0.9$, BS=100) | | | *71.17 ± 0.81%* | *19.15 ± 0.22%* | *4.41 ± 0.25%* | *56.42 ± 3.54%* |
| Retrained dense network* | | | *99.58 ± 0.04%* | *77.40 ± 0.37%* | *60.15 ± 0.00%* | *98.51 ± 0.53%* |

Table 6: Validation accuracies of a 89.26% sparse ResNet-34 ticket when trained on a TinyImageNet subset.

| Ticket parameters | | | Subset sizes | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | 1% | 2% | 5% | 10% |
| 0.0 | 100 | ✗ | 4.72 ± 0.46% | 7.69 ± 0.39% | 16.46 ± 0.24% | 27.34 ± 0.18% |
| 0.0 | 256 | | **10.49 ± 1.14%** | **17.90 ± 1.72%** | **29.51 ± 1.71%** | **37.87 ± 0.43%** |
| 0.9 | 100 | | 3.41 ± 0.28% | 5.18 ± 0.59% | 14.33 ± 0.55% | 8.36 ± 13.61% |
| 0.9 | 256 | | 3.40 ± 0.62% | 6.17 ± 0.85% | 11.42 ± 0.31% | 21.25 ± 1.63% |
| 0.0 | 100 | ✓ | 10.31 ± 1.19% | 15.29 ± 0.74% | 26.01 ± 1.19% | 36.35 ± 1.14% |
| 0.0 | 256 | | **15.94 ± 0.69%** | **22.90 ± 1.21%** | **34.18 ± 0.64%** | **43.40 ± 0.26%** |
| 0.9 | 100 | | 6.00 ± 0.58% | 8.76 ± 0.86% | 17.59 ± 1.09% | 26.50 ± 0.83% |
| 0.9 | 256 | | 4.95 ± 0.24% | 7.65 ± 0.78% | 15.31 ± 0.75% | 25.55 ± 0.61% |
| *Dense Network* | | | 4.32 ± 0.13% | 6.33 ± 0.19% | 13.19 ± 0.33% | 25.13 ± 0.57% |

the performance of the trained network. Results for tickets at 89.26% sparsity on TinyImageNet are listed in Table 6. Additional results and visualizations can be found in the appendix.

We notice that a significant portion of the validation accuracy attained with the full training dataset can be recovered in scenarios with higher SGD stability. In the most extreme case, we can recover $\sim 75\%$ validation accuracy of the full dataset by training with 50 randomly chosen images per class of TinyImageNet. This is a relative gain of $\sim 72\%$ over training the dense network with that subset. In the appendix, we show results for a more robust dataset subset selection criterion, for which the results follow a similar trend.

## 6 Discussion

### 6.1 Loss Basin interpretation

Paul et al. (2022) interprets stability to SGD noise as having two solutions that lie in the same loss basin. They observe that when the dense network is stable to SGD noise, then the derived tickets also lie in the same loss basin, with limited interpolation error between subsequent tickets. Fort et al. (2020) show that early in training the loss basin is determined, and that in the rest of training the optimizer will converge to

that loss basin. When using late rewinding in the LTH procedure to a stable set of weights, this means that in essence the loss basin is determined for all subsequent tickets.

**The impact of momentum.** The application of momentum allows for higher validation accuracy for the dense network, by compounding gradients over different minibatches. However, this clearly has an impact on the SGD stability of the network. We can see from the forgetting scores Table 1 that momentum causes a significant increase in forgetting events, meaning a more exploratory optimization path, and going through different loss basins. This finally results in a better loss basin that is found, as we have a higher validation accuracy for the model, but we hypothesize that this basin is much narrower than the basins found without momentum. As such, when pruning this loss basin becomes more difficult to find, leading to more significant accuracy degradation when coupled with the exploratory behaviour. In the appendix we show empirically that solutions with $\mu = 0.9$ take significantly longer training to select a loss basin.

**Small Batch Size.** When using a small batch size, the gradient of a single (or multiple) such samples, influences the total batch gradient in a much greater fashion. As such, the optimization process is influenced much more by the difficult samples and this can lead the model to better loss basins. When a lot of parameters are removed, it is more difficult for the model to fit the difficult samples, and this can cause accuracy loss.

**Limited training.** By limiting the training we also limit the exploration time. In the cases with momentum this means that rather than continuing to explore for the best possible loss basin, instead the model converges to a 'good enough' basin. This avoids issues where the loss basin is not attainable anymore after pruning. Conversely, in cases without momentum, the loss basin is attained early during training.

### 6.2 Usefulness of SGD instability

We argue that instability to SGD noise as it is currently defined based on the error barrier, is a corollary rather than a cause for the occurrence of winning tickets. We make this argument because we can see cases where one hyperparameter configuration outperforms significantly another at sparsity, even though the instability is still at the upper bound, meaning a maximum interpolation error at random chance. As such, instability is not enough to justify the (lack of) accuracy drop when pruning. A more useful metric would be to measure the number of training epochs needed to achieve this low error barrier as we calculate in the appendix, but this is significantly more expensive.

### 6.3 The emergence of useful features

We notice in several experiments that, when instability is low enough, the tickets found via IMP contain useful features without additional training. This suggests that by pruning repeatedly some information of the dataset is encoded within the mask, dependent on how low SGD instability is. This information can be used to recover a nontrivial percentage (more than 15% in the case of TinyImageNet) of the validation accuracy of a fully trained dense network, by finetuning a linear classifier on the features.

This effect has been shown to be independent of the application of late-rewinding – which encodes some information in the weights by virtue of pretraining. Additionally, the features are not overfitting the dataset, but can rather be transferred to other datasets, and can even outperform those encoded in dense networks trained on the source dataset in several cases.

This behavior is reminiscent of the Strong Lottery Ticket Hypothesis (Ramanujan et al., 2020), which poses that any trained network can be approximated by pruning connections from a sufficiently large untrained network. There is however a nuanced difference, in that the stable tickets found by our approach require a finetuned classification layer to achieve good validation accuracy.

## 7 Conclusion

In this research we study a number of training configurations and their influence on Lottery Tickets. We notice that commonly used hyperparameters, such a momentum and lower batch sizes have positive effect on the generalization of dense networks as already reported in the literature, but can negatively impact lottery

ticket performance. We have observed that while some links can be made to instability to SGD noise, this metric is insufficient to fully explain the differences created by these hyperparameters during the training process.

When applying these insights in training instability on the Lottery Ticket Hypothesis, we notice that in the cases with the least training instability, winning lottery tickets can be found at higher sparsities without any pretraining necessary. Even when considering additional stability in the form of pretraining, the more stable approaches show less accuracy degradation at higher sparsities.

On top of the higher validation accuracy at deeper sparsities, we show that tickets found with a higher stability exhibit several desirable properties encoded by the pruned initialization. We notice that these tickets achieve better few-shot generalizability, and can in fact be used as frozen feature extractors to a remarkable accuracy.

## 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.

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.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=dNigytemkL`.

Damien Ferbach, Baptiste Goujaud, Gauthier Gidel, and Aymeric Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pp. 3853–3861, 2024.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

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, 2020. ISBN 2640-3498.

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.

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.

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.

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

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

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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.

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. URL `https://openreview.net/forum?id=H1oyRlYgg`.

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

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H.S. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

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

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

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.

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.

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.

Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Mukund Varma T, Xuxi Chen, Zhenyu Zhang, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Sparse winning tickets are data-efficient image recognizers. In Alice H Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

Vincent Vanhoucke, Andrew Senior, Mark Z Mao, et al. Improving the speed of neural networks on cpus. In *Proc. deep learning and unsupervised feature learning NIPS workshop*, volume 1, pp. 4, 2011.

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.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 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.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4320–4328, 2018.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *Ninth International Conference on Learning Representations 2021*, 2021.

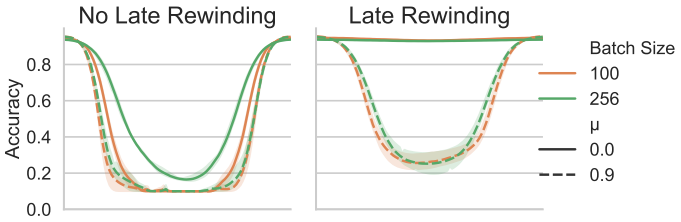
Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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.

## Appendices

- Appendix A contains additional results for experiments conducted in Section 4.1 of the main paper. More specifically interpolation curves for the different different dataset and network combinations, and visualizations of the Train Convergence at different sparsities for tickets are provided. To complement these results, we highlight the relationship between forgetting events and Linear Mode Connectivity in Appendix A.1.
- Appendix B contains the full results for the experiments conducted in Section 4.2. This includes visualizations of the evolution of ticket validation accuracy for each setting, as well as tabular representations for the ResNet-18 + {CIFAR-10, CIFAR-100} setting. Additionally, more extreme batch sizes for CIFAR-10 are explored in Appendix B.1, and the optimal rewind points for different configurations are calculated in Appendix B.2.
- Appendix C contains additional results for the experiments conducted in Section 5.1. More specifically, we show transferability for other datasets, and visualize the evolution of probe accuracy in function of sparsity and network depth for the different settings.
- Appendix D contains additional results for the experiments conducted in Section 4.3. We show additional results on ResNet-18 + CIFAR-10 for the same reduced computational budgets.
- Appendix E contains additional results for the experiments conducted in Section 5.2. These include results for other settings, more extreme dataset subsets for CIFAR-10, and another subset selection criterion by using sample difficulty from Toneva et al. (2019).
- Appendix F contains the hyperparameter configurations used in the experiments.

## A Additional Training Stability Results

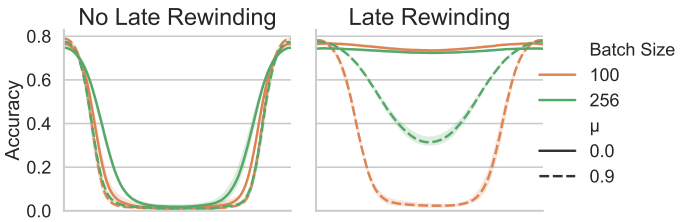



(a)

| | | Error Barrier | |
|---|---|---|---|
| $\mu$ | Batch | w/o late rewinding | w/ late rewinding |
| 0.0 | 100 | 84.58 ± 0.00 | 1.45 ± 0.17 |
| 0.0 | 256 | 77.53 ± 0.73 | 0.86 ± 0.05 |
| 0.9 | 100 | 85.27 ± 0.04 | 69.98 ± 3.55 |
| 0.9 | 256 | 85.34 ± 0.04 | 70.34 ± 4.73 |

(b)

Figure 2: Instability for a dense **ResNet-18** model trained on **CIFAR10** with different training configurations. We show the full interpolation curve in **(a)** and the error barrier in **(b)**. Notice that the interpolation error with $\mu = 0.0$, 256 BS is higher than random chance, which we do not observe for other dataset & network combinations.




(a)

| | | Error Barrier | |
|---|---|---|---|
| $\mu$ | Batch | w/o late rewinding | w/ late rewinding |
| 0.0 | 100 | 75.19 ± 0.26 | 2.98 ± 0.10 |
| 0.0 | 256 | 72.79 ± 0.53 | 1.95 ± 0.30 |
| 0.9 | 100 | 77.97 ± 0.07 | 75.93 ± 0.45 |
| 0.9 | 256 | 76.43 ± 0.05 | 46.15 ± 1.66 |

(b)

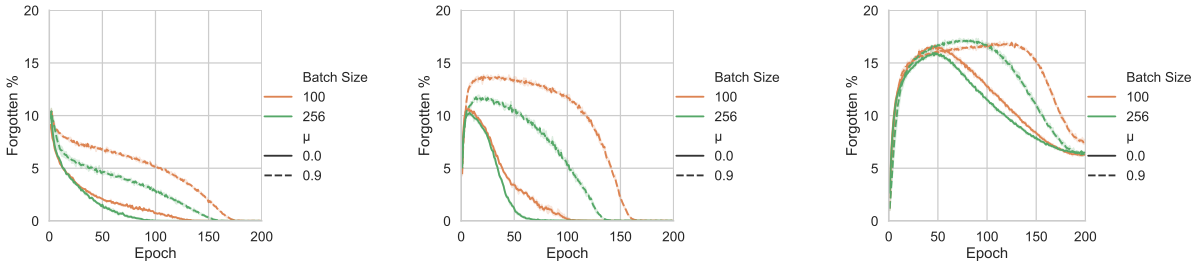
Figure 3: Instability for a dense **ResNet-18** model trained on **CIFAR100** with different training configurations. We show the full interpolation curve in **(a)** and the error barrier in **(b)**.

Table 7: Average forgetting events per sample, proportion of unforgettable samples and convergence statistics for different hyperparameter configurations used in training **ResNet-18** models on **CIFAR-10**.

| $\mu$ | Batch Size | Forgetting events | Unforgettable % | Convergence |
|---|---|---|---|---|
| 0.0 | 100 | 5.73 ± 8.26 | 42.90 ± 0.16% | 0.960 ± 0.001 |
| 0.0 | 256 | 3.35 ± 4.44 | 42.18 ± 1.18% | 0.939 ± 0.003 |
| 0.9 | 100 | 13.43 ± 15.79 | 22.92 ± 0.50% | 0.903 ± 0.003 |
| 0.9 | 256 | 8.61 ± 11.23 | 27.68 ± 3.05% | 0.927 ± 0.005 |

Table 8: Average forgetting events per sample, proportion of unforgettable samples and convergence statistics for different hyperparameter configurations used in training **ResNet-18** models on **CIFAR-100**.

| $\mu$ | Batch Size | Forgetting events | Unforgettable % | Convergence |
|---|---|---|---|---|
| 0.0 | 100 | 6.78 ± 5.44 | 16.21 ± 0.19% | 0.927 ± 0.002 |
| 0.0 | 256 | 5.10 ± 4.06 | 17.15 ± 0.23% | 0.905 ± 0.003 |
| 0.9 | 100 | 25.46 ± 15.62 | 4.59 ± 0.15% | 0.814 ± 0.001 |
| 0.9 | 256 | 15.77 ± 11.57 | 8.78 ± 0.20% | 0.862 ± 0.004 |




(a) (b) (c)

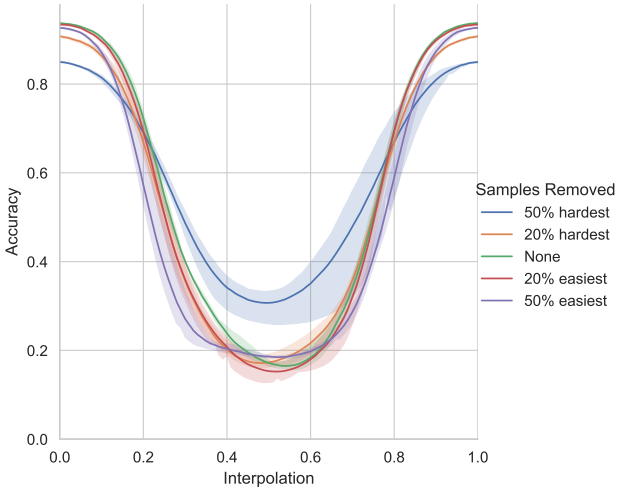
Figure 4: The evolution of forgetting events during training for **(a)** ResNet-18 + CIFAR-10, **(b)** ResNet-18 + CIFAR-100, and **(c)** ResNet-34 + TinyImageNet given different hyperparameter configurations. Notice that approaches with $\mu = 0.9$ consistently have significantly more forgetting events late in training.

### A.1 SGD instability and forgetting events

As shown in Toneva et al. (2019), a subset of the CIFAR-10 dataset, selected by removing samples with a low forgetting score, can be used to train a ResNet-18 model without loss of generalization, as compared to the full dataset. The authors show that this is true for removing up to the 30% of the samples with the lowest forgetting events. This indicates that the samples with more forgetting events are more critical for the model training. We hypothesize that these samples, while more critical to the performance, also serve as a factor of SGD noise. To verify this, we generate dataset subsets where we remove the samples with the least forgetting events (termed 'easiest'), and datasets where we remove the samples with the most forgetting events (termed 'hardest'). For each subset, we train a ResNet-18 model on CIFAR-10 with $\mu = 0.0$, BS = 256 for three random seeds. We then show the interpolation curves, error barriers and validation accuracies for each dataset subset in Figure 5.

From these results, we can confirm that removing the least-forgotten samples has no significant impact on the validation accuracy, but that removing the most-forgotten samples significantly impacts the generalization error. When analyzing the instability to SGD noise, we notice a decrease in instability when removing harder

samples. Removing an equivalent amount of easy samples has no such significant impact. If anything, we can notice a slight worsening in the interpolation curve, as the error barrier is reached earlier during interpolation.




(a)

| | Easiest samples removed | | Hardest samples removed | |
|---|---|---|---|---|
| Original | 20% | 50% | 20% | 50% |
| 77.53 | 78.36 | 74.38 | 73.68 | 54.36 |

(b)

| | Easiest samples removed | | Hardest samples removed | |
|---|---|---|---|---|
| Original | 20% | 50% | 20% | 50% |
| 93.63% | 93.40% | 92.66% | 90.76% | 84.96% |

(c)

Figure 5: **(a)** Linear interpolation interpolation curves, **(b)** error barriers, and **(c)** validation accuracies of **ResNet18** networks trained on different subsets of **CIFAR10**. The networks are trained with $\mu = 0.0$, 256 batch size.

## B Full Lottery Ticket results

Table 9: Lottery Ticket extracted from **ResNet-18** on **CIFAR-10 / CIFAR-100** performances for different training parameters at different sparsity levels.

| Ticket Parameters | | | | Sparsity | | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | $\mu$ | Batch Size | Pretrained | 0.00% | 67.23% | 89.26% | 96.48% | 98.85% |
| CIFAR-10 | 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% |
| CIFAR-100 | 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% |

In Figures 6 to 8 we plot the evolution of Lottery Ticket validation accuracy at all studied sparsities. These graphs serve to complement the results from Table 2 in the main paper and Table 9 above. In those tables we highlight accuracies at specific sparsities, which are indicated on the x-axis in the figures. We can see more clearly in these graphs that the more unstable approaches suffer from more significant accuracy degradation throughout the pruning process, even though the dense networks attained with those procedures have superior performances.

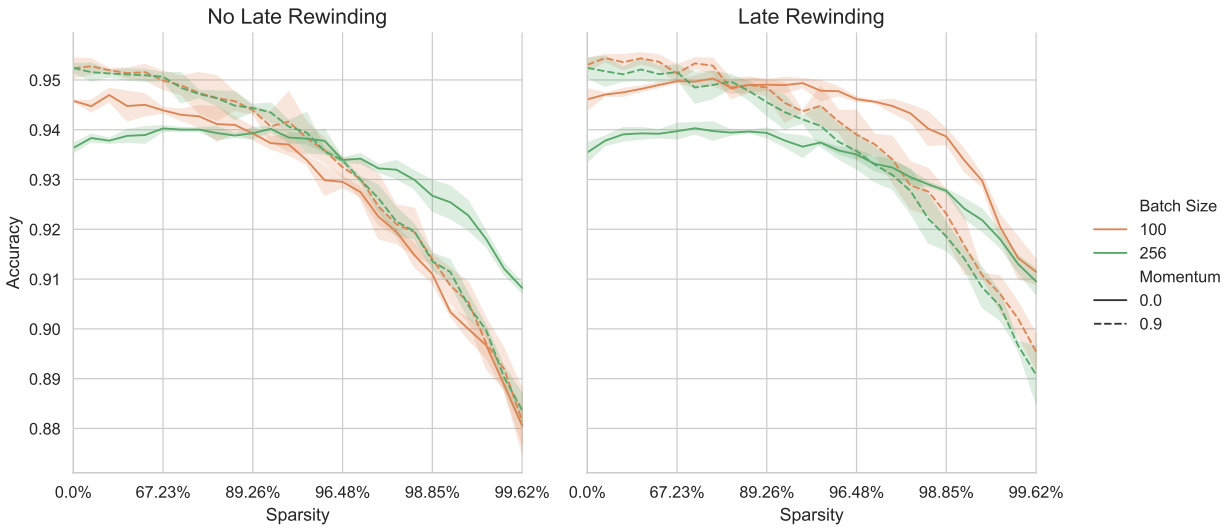



Figure 6: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-18** model on the **CIFAR-10** dataset.

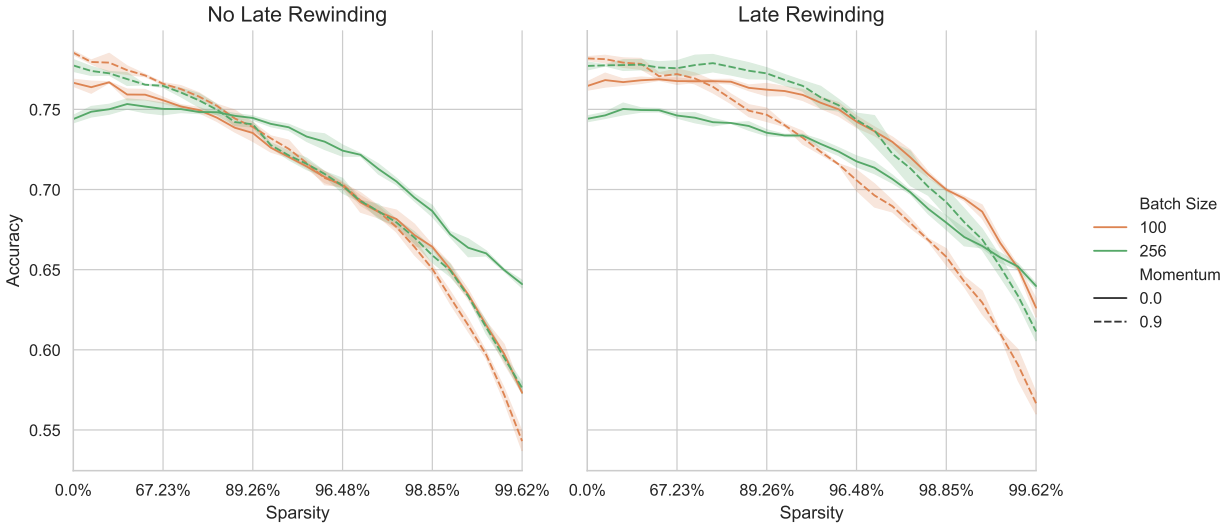



Figure 7: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-18** model on the **CIFAR-100** dataset.

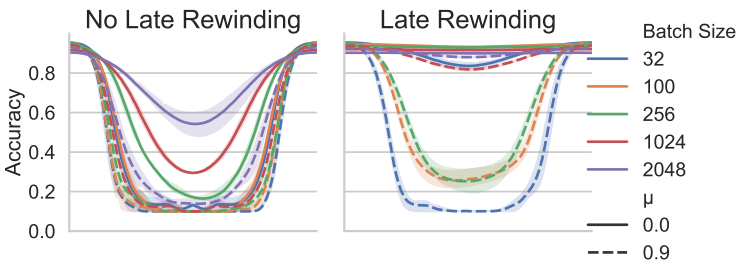
Figure 8: Validation accuracies at different sparsities for Lottery Tickets extracted from a **ResNet-34** model on the **TinyImageNet** dataset. Notice that some iterations record a significant accuracy drop, together with a high standard deviation. This indicates that at that iteration one (or more) random runs did not train and instead remained at random chance (0.5%).

### B.1 More extreme batch sizes

In the main paper, we have demonstrated that lower batch sizes result in more SGD instability, but better generalization of the dense network. Conversely, for higher batch sizes, we see a reduction in SGD instability, which allows for winning Lottery Tickets to be found without pretraining.

In Table 10, we cover a wider variety of batch sizes for ResNet-18 + CIFAR-10 than in Appendix B, 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. Interestingly, we observe cases where the accuracy might suffer from too much stability. For example, comparing $\mu = 0.9$, BS=2048 in the settings with and without late rewinding, we see that there is a non-trivial accuracy drop at higher sparsities in the setting with late rewinding compared to that without late rewinding.

In Figure 9 we clearly see the effect of batch size on instability. We notice that at very large batch sizes, the application of momentum in combination with late rewinding still results in almost stable configurations.




(a)

| | | Error Barrier | |
|---|---|---|---|
| $\mu$ | Batch | w/o late rewinding | w/ late rewinding |
| 0.0 | 32 | 85.40 ± 0.00 | 11.85 ± 1.32 |
| | 100 | 84.58 ± 0.00 | 1.45 ± 0.17 |
| | 256 | 77.53 ± 0.73 | 0.86 ± 0.05 |
| | 1024 | 62.42 ± 0.00 | 0.55 ± 0.06 |
| | 2048 | 36.24 ± 4.84 | 0.30 ± 0.16 |
| 0.9 | 32 | 84.71 ± 0.08 | 84.77 ± 0.06 |
| | 100 | 85.27 ± 0.04 | 69.98 ± 3.55 |
| | 256 | 85.34 ± 0.04 | 70.34 ± 4.73 |
| | 1024 | 84.29 ± 0.00 | 12.03 ± 0.48 |
| | 2048 | 79.35 ± 2.03 | 4.65 ± 0.17 |

(b)

Figure 9: Instability for a dense **ResNet-18** model trained on **CIFAR10** with different training configurations, featuring more batch sizes. We show the full interpolation curve in **(a)** and the error barrier in **(b)**.

Table 10: Validation accuracy of **ResNet-18** lottery tickets extracted for **CIFAR-10** given more extreme batch sizes at different sparsity levels. CAUTION: these results were attained with only a single run each.

| Ticket Parameters | | | Sparsity | | | | |
|---|---|---|---|---|---|---|---|
| $\mu$ | Pretrain | Batch Size | 0.00% | 67.23% | 89.26% | 96.48% | 98.85% |
| 0.0 | ✗ | 8 | 95.29 ± 0.00% | 95.54 ± 0.00% | 94.81 ± 0.00% | 93.79 ± 0.00% | 91.93 ± 0.00% |
| | | 32 | 95.41 ± 0.00% | 95.29 ± 0.00% | 94.71 ± 0.00% | 93.68 ± 0.00% | 91.72 ± 0.00% |
| | | 1024 | 91.87 ± 0.00% | 91.81 ± 0.00% | 91.62 ± 0.00% | 91.24 ± 0.00% | 89.86 ± 0.00% |
| | | 2048 | 90.92 ± 0.00% | 90.74 ± 0.00% | 90.37 ± 0.00% | 90.00 ± 0.00% | 88.78 ± 0.00% |
| 0.9 | ✗ | 8 | 92.92 ± 0.00% | 88.31 ± 0.00% | 77.92 ± 0.00% | 70.27 ± 0.00% | 10.00 ± 0.00% |
| | | 32 | 94.53 ± 0.00% | 93.75 ± 0.00% | 91.49 ± 0.00% | 90.02 ± 0.00% | 88.62 ± 0.00% |
| | | 1024 | 94.03 ± 0.00% | 94.30 ± 0.00% | 93.54 ± 0.00% | 92.80 ± 0.00% | 90.57 ± 0.00% |
| | | 2048 | 93.14 ± 0.00% | 93.46 ± 0.00% | 93.60 ± 0.00% | 93.54 ± 0.00% | 92.31 ± 0.00% |
| 0.0 | ✓ | 8 | 95.28 ± 0.00% | 95.24 ± 0.00% | 95.54 ± 0.00% | 94.86 ± 0.00% | 93.34 ± 0.00% |
| | | 32 | 95.56 ± 0.00% | 95.39 ± 0.00% | 95.52 ± 0.00% | 94.78 ± 0.00% | 93.38 ± 0.00% |
| | | 1024 | 92.12 ± 0.00% | 91.92 ± 0.00% | 91.70 ± 0.00% | 91.60 ± 0.00% | 89.87 ± 0.00% |
| | | 2048 | 90.19 ± 0.00% | 90.27 ± 0.00% | 90.33 ± 0.00% | 90.29 ± 0.00% | 88.17 ± 0.00% |
| 0.9 | ✓ | 8 | 92.31 ± 0.00% | 90.93 ± 0.00% | 88.98 ± 0.00% | 87.20 ± 0.00% | 63.56 ± 0.00% |
| | | 32 | 94.57 ± 0.00% | 93.74 ± 0.00% | 93.08 ± 0.00% | 91.63 ± 0.00% | 89.78 ± 0.00% |
| | | 1024 | 94.22 ± 0.00% | 93.89 ± 0.00% | 93.96 ± 0.00% | 93.22 ± 0.00% | 91.69 ± 0.00% |
| | | 2048 | 92.90 ± 0.00% | 92.92 ± 0.00% | 93.05 ± 0.00% | 92.47 ± 0.00% | 90.91 ± 0.00% |

### B.2 Finding optimal rewind points

For several parameter configurations of ResNet-18 + CIFAR-10, the default rewind point of 2 epochs was insufficient to provide stability to SGD noise. As such, we conduct a study to determine at which points (measured in epochs) SGD stability emerges. We follow the same definition of stability as in Frankle et al. (2020), meaning if the error across the linear path is less than 2%, the weights are considered stable to SGD noise. Note that this definition assumes an absolute difference of 2%, not relative difference to the accuracies.

To speed up this procedure, we use a binary search algorithm to determine the first stable rewind point for each configuration. For this algorithm to work, we assume that stability to SGD noise increases monotonic, which we have validated experimentally. This allows us to find this rewind point in $\lceil log_2 200 \rceil + 1 = 9$ steps. We employ three random initializations for the rewind point, and to measure stability we use three different sets of SGD noise, for a total of 9 runs per configuration. In total this results in $\sim 81$ full training runs for each configuration.

We list the results in Table 11. We notice that, as discussed in the main paper, both momentum and lower batch size impact stability negatively. In particular, the use of momentum has a significant impact, where we notice that in those cases stability only emerges at the earliest at around 10% of the total training epochs. In the worst case scenario, stability is only achieved at around 23% of the total training time.

Table 11: The first epoch at which stability to SGD noise occurs for the considered model and dataset combinations trained under different hyperparameter settings.

| | | $\mu = 0.0$ | | $\mu = 0.9$ | |
|---|---|---|---|---|---|
| Network | Dataset | 100 BS | 256 BS | 100 BS | 256 BS |
| ResNet-18 | CIFAR-10 | 2.3 ± 0.5 | 1.7 ± 0.5 | 42.7 ± 1.9 | 23.7 ± 1.2 |
| ResNet-18 | CIFAR-100 | 3.0 ± 0.0 | 2.0 ± 0.0 | 47.3 ± 2.1 | 19.0 ± 0.8 |
| ResNet-34 | TinyImageNet | 5.0 ± 0.0 | 3.7 ± 0.9 | 36.7 ± 1.9 | 18.0 ± 1.4 |

# C Additional feature emergence results

## C.1 Linear Probing

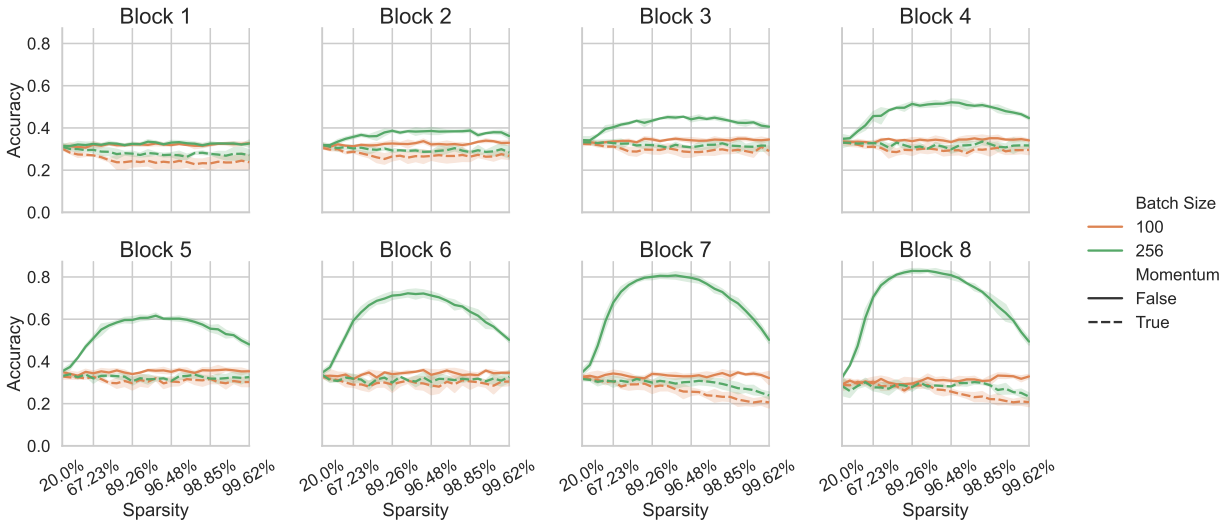
In Figures 10 to 12 we visualize the probes at each sparsity to complement the tabular results of Table 4 in the main paper and Tables 12 and 13 below. We can see that the selection of useful features evolves throughout the pruning process. While the features are all similar in the randomly initialized dense network, during the first few pruning iterations the most stable approaches show a steady increase in feature performance, while no evolution is noticeable in the most unstable approaches. Furthermore, we can see that those more stable approaches also have higher quality features in the deeper layers, which is not the case for those that are more unstable.

Table 12: Linear probing results at different locations in a **ResNet-18** ticket extracted for **CIFAR-10** under different training configurations. Note that unlike the ResNet-34 + TinyImageNet results in the main paper, we can achieve a good feature extractor without late rewinding, namely for $\mu = 0.0$, batch size = 256, which coincidentally has better instability than the other configurations tested, thus reinforcing our belief.

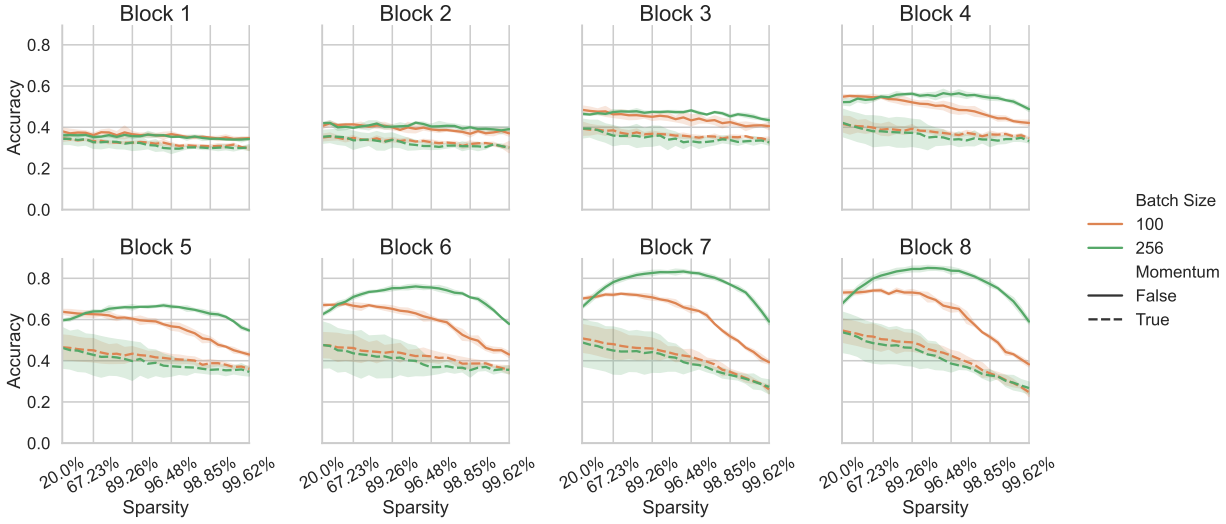
| Ticket parameters | | | 89.26% sparsity | | 96.48% sparsity | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | Block 4 | Block 8 | Block 4 | Block 8 |
| 0.0 | 100 | ✗ | 33.61 ± 1.60% | 30.93 ± 0.82% | 33.45 ± 0.41% | 30.51 ± 0.38% |
| 0.0 | 256 | | **51.38 ± 0.81%** | **82.87 ± 0.84%** | **52.17 ± 1.22%** | **80.85 ± 1.57%** |
| 0.9 | 100 | | 29.48 ± 2.03% | 28.65 ± 0.92% | 29.39 ± 1.81% | 25.71 ± 1.03% |
| 0.9 | 256 | | 30.70 ± 0.52% | 27.83 ± 1.55% | 29.87 ± 1.42% | 28.15 ± 0.21% |
| 0.0 | 100 | ✓ | 52.08 ± 0.95% | 73.11 ± 1.16% | 49.32 ± 1.95% | 65.81 ± 1.77% |
| 0.0 | 256 | | **56.27 ± 0.48%** | **84.53 ± 1.07%** | **55.91 ± 0.69%** | **83.71 ± 1.35%** |
| 0.9 | 100 | | 39.27 ± 1.58% | 48.92 ± 3.44% | 36.74 ± 0.67% | 42.49 ± 2.03% |
| 0.9 | 256 | | 37.27 ± 5.31% | 46.29 ± 7.13% | 33.97 ± 1.76% | 38.62 ± 3.30% |
| Permuted ($\mu = 0.0$, BS=256) | | | 32.57 ± 1.68% | 33.69 ± 0.56% | 29.88 ± 1.36% | 31.81 ± 0.61% |

Table 13: Linear probing results at different locations in a **ResNet-18** ticket extracted for **CIFAR-100** under different training configurations. Note that unlike the ResNet-34 + TinyImageNet results in the main paper, we can achieve a good feature extractor without late rewinding, namely for $\mu = 0.0$, batch size = 256, which coincidentally has less instability than the other configurations tested, thus reinforcing our belief.

| Ticket parameters | | | 89.26% sparsity | | 96.48% sparsity | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | Block 4 | Block 8 | Block 4 | Block 8 |
| 0.0 | 100 | ✗ | 9.50 ± 0.21% | 9.31 ± 0.76% | 9.82 ± 0.45% | 10.51 ± 0.32% |
| 0.0 | 256 | | **15.19 ± 0.41%** | **44.26 ± 0.88%** | **15.48 ± 0.35%** | **36.57 ± 0.87%** |
| 0.9 | 100 | | 7.13 ± 0.87% | 7.14 ± 0.27% | 7.29 ± 0.94% | 7.41 ± 1.54% |
| 0.9 | 256 | | 8.22 ± 0.09% | 7.84 ± 0.08% | 8.73 ± 0.43% | 8.38 ± 0.62% |
| 0.0 | 100 | ✓ | 16.02 ± 0.21% | 34.57 ± 0.48% | 14.32 ± 0.75% | 22.01 ± 1.17% |
| 0.0 | 256 | | **18.40 ± 0.59%** | **49.56 ± 0.85%** | **17.52 ± 0.66%** | **43.71 ± 1.19%** |
| 0.9 | 100 | | 8.63 ± 0.16% | 14.37 ± 1.17% | 8.43 ± 0.61% | 11.11 ± 0.28% |
| 0.9 | 256 | | 11.54 ± 1.21% | 21.07 ± 2.74% | 11.01 ± 1.30% | 16.18 ± 2.03% |
| Permuted ($\mu = 0.0$, BS=256) | | | 8.43 ± 0.03% | 10.13 ± 0.61% | 7.18 ± 0.34% | 4.88 ± 0.18% |

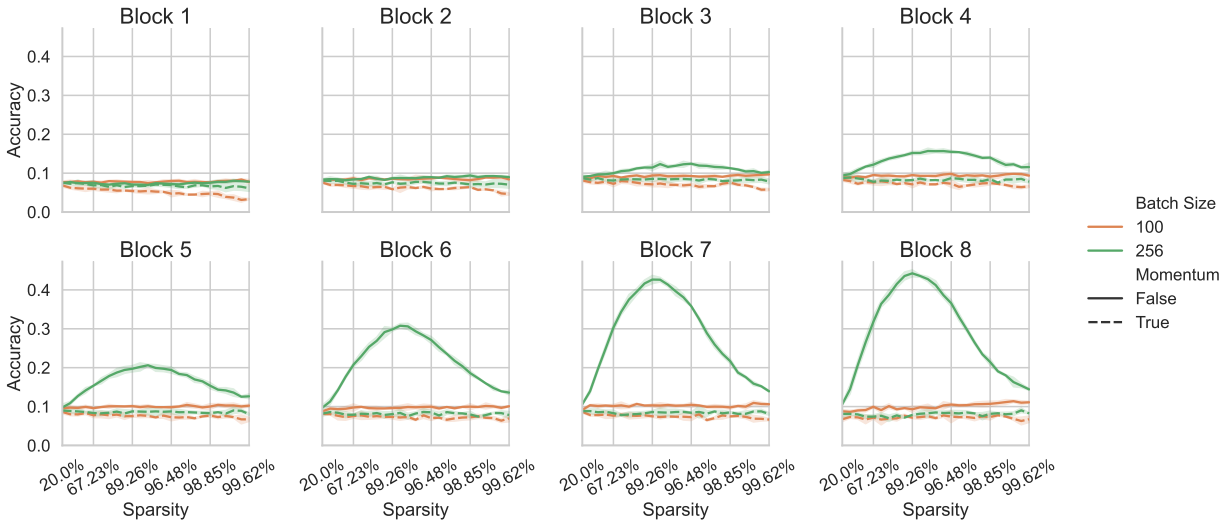



(a)

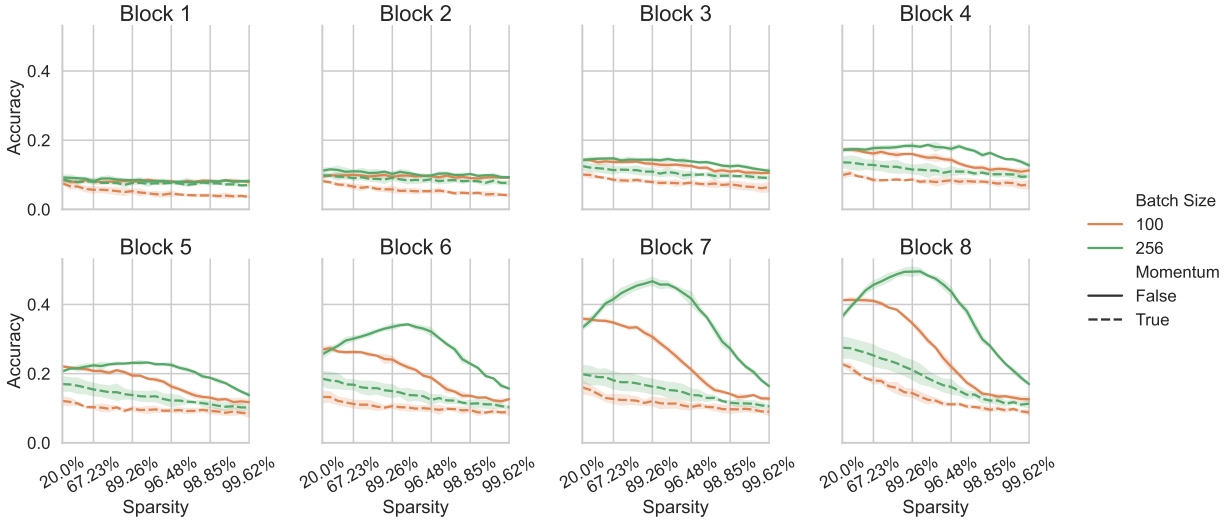



(b)

Figure 10: Linear probing results at all sparsities for **ResNet-18** Lottery Tickets on **CIFAR-10** given different hyperparameter configurations. **(a)** Without late rewinding, and **(b)** with late rewinding.

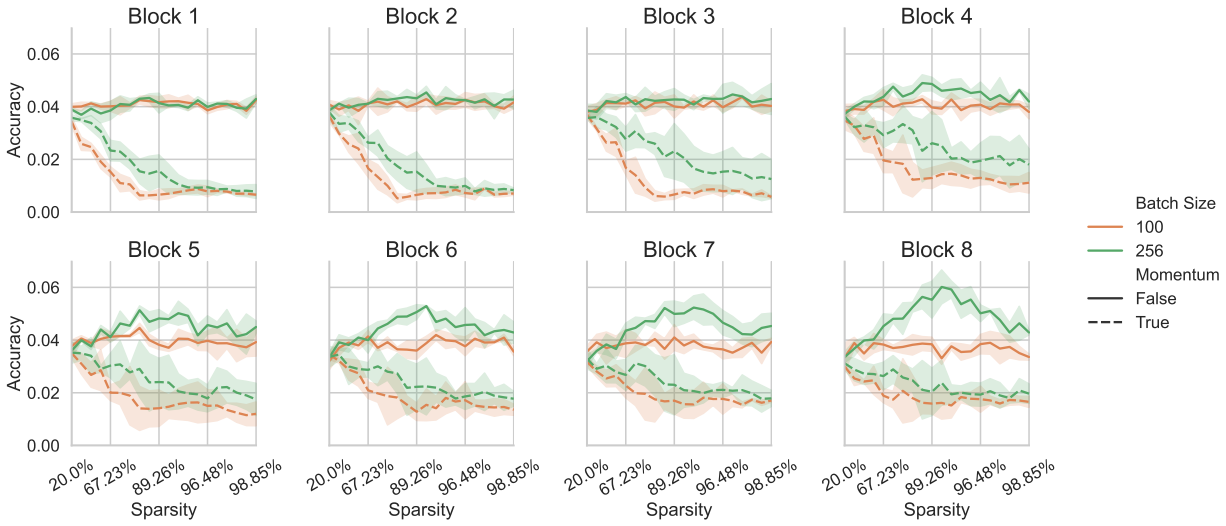



(a)

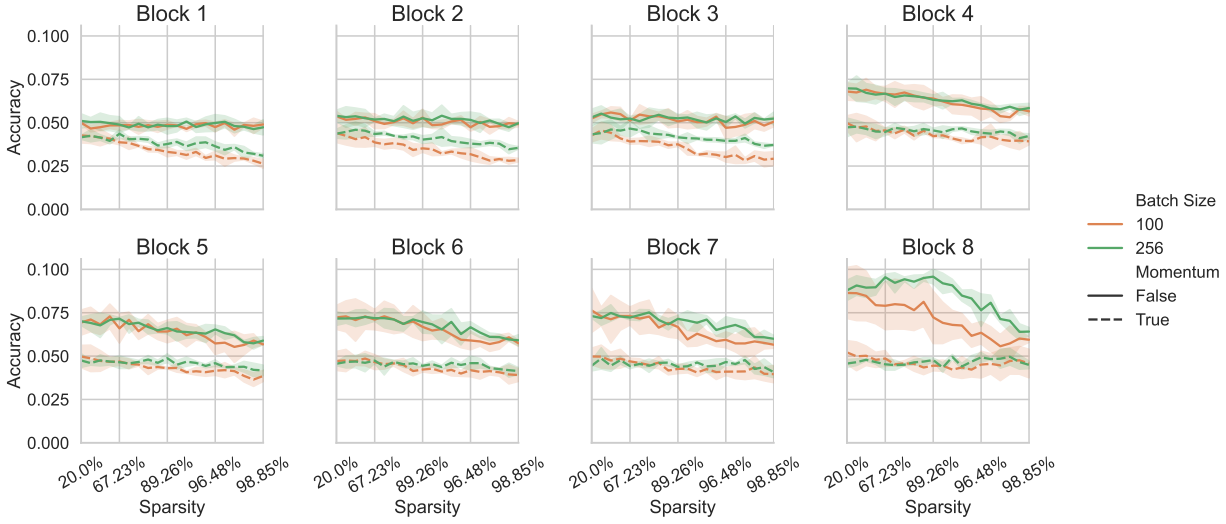



(b)

Figure 11: Linear probing results at all sparsities for **ResNet-18** Lottery Tickets on **CIFAR-100** given different hyperparameter configurations. **(a)** Without late rewinding, and **(b)** with late rewinding.




(a)




(b)

Figure 12: Linear probing results at all sparsities for **ResNet-34** Lottery Tickets on **TinyImageNet** given different hyperparameter configurations. **(a)** Without late rewinding, and **(b)** with late rewinding.

### C.2 Transferability

In Tables 14 and 15 we show the transferability results for frozen tickets extracted for CIFAR-100 and TInyImageNet. When listing the datasets in order of increasing complexity, we can achieve the following list: MNIST $<$ EuroSAT $\approx$ CIFAR-10 $<$ CIFAR-100 $<$ TinyImageNet. We notice in some experiments that our frozen tickets at initialization don't outperform any of the frozen fully trained dense networks. More specifically, in most of the cases where we transfer from a more complex dataset to a less complex dataset, the trained dense network outperforms the best pruned ticket by some margin. Additionally, transferring from frozen CIFAR-100 tickets seems better than transferring from frozen TinyImageNet tickets, either because the CIFAR-100 tickets have less instability, or due to the network structure.

Table 14: Transferability of frozen 89.26% **ResNet-18** tickets extracted on **CIFAR-100** with different configurations. Starred(*) entries correspond to dense trained networks, meaning the feature extractors have a sparsity of 0.00%.

| Ticket parameters | | | Target dataset | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | MNIST | CIFAR-10 | TinyImageNet | EuroSAT |
| 0.0 | 100 | ✗ | 84.47 ± 0.80% | 38.17 ± 0.23% | 8.74 ± 0.31% | 73.05 ± 1.00% |
| 0.0 | 256 | | **97.18 ± 0.20%** | **67.60 ± 0.89%** | **20.59 ± 0.97%** | **86.59 ± 0.54%** |
| 0.9 | 100 | | 79.71 ± 0.79% | 34.95 ± 1.95% | 6.78 ± 0.07% | 66.08 ± 1.99% |
| 0.9 | 256 | | 79.34 ± 3.08% | 35.84 ± 0.97% | 7.29 ± 0.61% | 68.76 ± 0.70% |
| 0.0 | 100 | ✓ | 96.83 ± 0.07% | 65.44 ± 1.14% | 22.81 ± 1.07% | 86.67 ± 1.10% |
| 0.0 | 256 | | **97.53 ± 0.18%** | **70.40 ± 0.48%** | **24.62 ± 0.51%** | **89.03 ± 0.09%** |
| 0.9 | 100 | | 91.92 ± 0.58% | 47.79 ± 1.06% | 11.27 ± 1.61% | 74.14 ± 1.63% |
| 0.9 | 256 | | 95.47 ± 0.46% | 53.86 ± 2.86% | 16.44 ± 1.08% | 80.78 ± 1.61% |
| Frozen dense* ($\mu = 0.0$, BS=256) | | | *95.83 ± 0.24%* | *78.12 ± 0.11%* | *17.10 ± 0.97%* | *79.14 ± 2.16%* |
| Frozen dense* ($\mu = 0.9$, BS=100) | | | *90.10 ± 0.36%* | *75.39 ± 0.41%* | *13.16 ± 0.46%* | *67.04 ± 3.70%* |

Table 15: Transferability of a frozen 89.26% **ResNet-34** tickets extracted on **TinyImageNet** with different configurations. Starred(*) entries correspond to dense trained networks, meaning the feature extractors have a sparsity of 0.00%.

| Ticket parameters | | | Target dataset | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | MNIST | CIFAR-10 | CIFAR-100 | EuroSAT |
| 0.0 | 100 | ✗ | 66.05 ± 1.23% | 27.87 ± 0.60% | 10.25 ± 0.63% | 63.25 ± 0.76% |
| 0.0 | 256 | | **92.29 ± 2.16%** | **46.72 ± 1.98%** | **22.86 ± 1.06%** | **78.19 ± 0.30%** |
| 0.9 | 100 | | 62.50 ± 11.85% | 26.54 ± 1.83% | 6.98 ± 1.12% | 46.19 ± 8.49% |
| 0.9 | 256 | | 67.25 ± 4.44% | 26.83 ± 2.69% | 8.87 ± 1.71% | 54.25 ± 5.52% |
| 0.0 | 100 | ✓ | 93.38 ± 1.46% | 50.19 ± 3.07% | 26.99 ± 2.08% | 80.38 ± 1.01% |
| 0.0 | 256 | | **96.56 ± 0.45%** | **59.00 ± 1.22%** | **35.73 ± 2.02%** | **87.88 ± 1.04%** |
| 0.9 | 100 | | 85.83 ± 3.89% | 40.15 ± 1.69% | 13.60 ± 1.41% | 68.28 ± 0.63% |
| 0.9 | 256 | | 86.63 ± 5.18% | 38.01 ± 2.34% | 13.23 ± 1.79% | 68.76 ± 3.24% |
| Frozen dense* ($\mu = 0.0$, BS=256) | | | *92.77 ± 0.00%* | *62.58 ± 0.09%* | *40.91 ± 0.46%* | *90.10 ± 0.45%* |
| Frozen dense* ($\mu = 0.9$, BS=100) | | | *92.79 ± 0.56%* | *65.11 ± 0.19%* | *44.24 ± 0.02%* | *90.28 ± 0.52%* |

## D Additional results for limited Mask Search Budgets

Table 16: AIMP results for different hyperparameter settings of **ResNet-18** on **CIFAR-10**.

| Ticket Parameters | | | | Sparsity | | | |
|---|---|---|---|---|---|---|---|
| Budget | $\mu$ | Batch Size | Pretrained | 67.23% | 89.26% | 96.48% | 98.85% |
| 20eps | 0.0 | 100 | ✗ | 94.34% | 93.76% | 92.65% | 90.26% |
| | 0.0 | 256 | | 93.30% | 92.69% | 91.57% | 88.34% |
| | 0.9 | 100 | | **95.18%** | **94.82%** | **93.31%** | **91.75%** |
| | 0.9 | 256 | | 95.01% | 94.55% | 93.25% | 91.18% |
| 50eps | 0.0 | 100 | ✗ | 94.31% | 93.66% | 92.57% | 90.70% |
| | 0.0 | 256 | | 93.71% | 93.38% | 93.08% | 91.50% |
| | 0.9 | 100 | | 95.07% | 94.24% | 92.74% | 90.83% |
| | 0.9 | 256 | | **95.14%** | **94.69%** | **93.65%** | **91.90%** |

## E Additional few-shot learning results

In Figures 13 to 15 we visualize the full few-shot results at all sparsities to complement the tabular results recorded in Table 6 in the main paper and Tables 17 and 18 below. As with the linear probing results, we notice that the initial dense network has limited propensity for few-shot learning, but this increases for the more stable approaches during the initial few pruning iterations. This shows that the better few-shot performance is associated with the linear probing efficacy. Indeed, it is quite logical to assume that if good features are already encoded in the network, the network has a 'headstart' for training with limited data, and consequently can achieve better accuracy with limited samples.

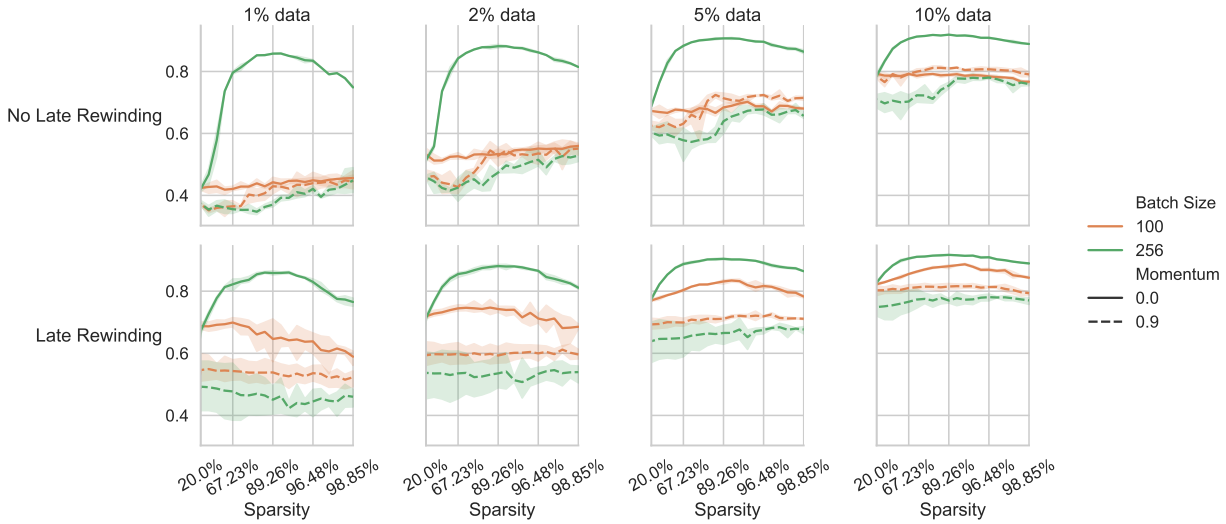



Figure 13: Few-shot performances at all sparsities for **ResNet-18** Lottery Tickets on **CIFAR-10** given different hyperparameter configurations.

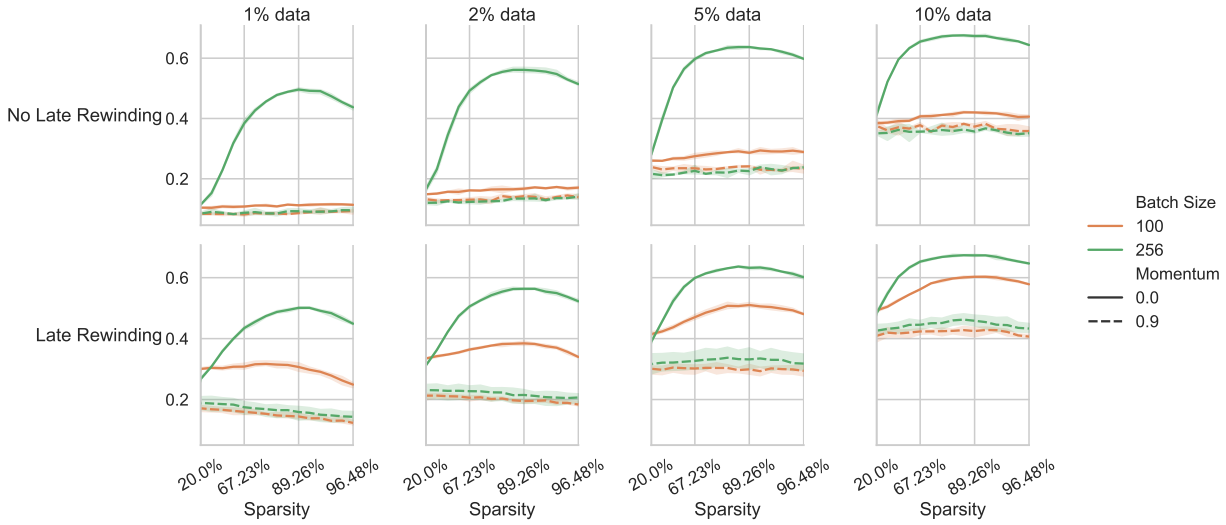



Figure 14: Few-shot performances at all sparsities for **ResNet-18** Lottery Tickets on **CIFAR-100** given different hyperparameter configurations.

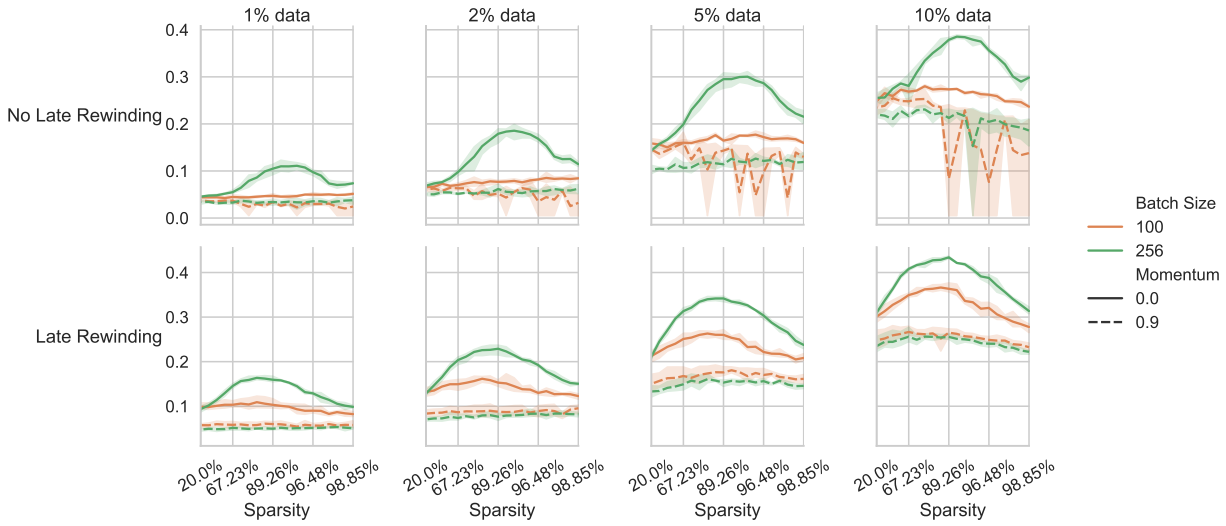



Figure 15: Few-shot performances at all sparsities for **ResNet-34** Lottery Tickets on **TinyImageNet** given different hyperparameter configurations.

Table 17: Validation accuracies of a 89.26% sparse **ResNet-10** ticket when trained on a **CIFAR-10** subset.

| Ticket parameters | | | Subset sizes | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Batch Size | Pretrained | 1% | 2% | 5% | 10% |
| 0.0 | 100 | ✗ | 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% |
| 0.0 | 100 | ✓ | 68.02 ± 2.81% | 74.49 ± 0.60% | 83.43 ± 0.25% | 88.69 ± 0.48% |
| 0.0 | 256 | | **85.84 ± 0.70%** | **88.09 ± 0.47%** | **90.40 ± 0.21%** | **91.68 ± 0.11%** |
| 0.9 | 100 | | 53.85 ± 3.47% | 59.29 ± 2.70% | 71.10 ± 0.55% | 81.51 ± 0.90% |
| 0.9 | 256 | | 45.06 ± 4.52% | 53.55 ± 5.55% | 66.54 ± 0.32% | 76.99 ± 0.75% |
| *Dense Network* | | | 42.54 ± 1.15% | 51.25 ± 0.78% | 67.15 ± 0.34% | 78.42 ± 0.53% |

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

| Ticket parameters | | | Subset sizes | | | |
|---|---|---|---|---|---|---|
| $\mu$ | Pretraining | Batch Size | 1% | 2% | 5% | 10% |
| 0.0 | 100 | ✗ | 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% |
| 0.0 | 100 | ✓ | 30.69 ± 1.38% | 38.44 ± 0.70% | 51.03 ± 0.72% | 60.29 ± 0.18% |
| 0.0 | 256 | | **50.10 ± 0.39%** | **56.36 ± 0.31%** | **63.23 ± 0.50%** | **67.32 ± 0.35%** |
| 0.9 | 100 | | 14.44 ± 0.97% | 19.51 ± 0.61% | 29.91 ± 1.60% | 42.48 ± 1.64% |
| 0.9 | 256 | | 15.91 ± 2.34% | 21.50 ± 2.43% | 33.17 ± 3.51% | 45.86 ± 1.76% |
| *Dense Network* | | | 9.44 ± 0.48% | 13.69 ± 0.45% | 22.73 ± 0.09% | 34.34 ± 0.48% |

### E.1 Using forgetting scores as selection criterion

The results earlier in this section, and in Section 5.2 in the main paper are achieved with randomly selected class-balanced subsets from the training dataset. In this experiment we study a more robust selection criterion by using sample-wise forgetting statistics from Toneva et al. (2019) to generate class-balanced dataset subsets. We show the results for CIFAR-10 subsets on the 89.26% sparse ticket achieved with $\mu = 0.0$, batch size 256 in Table 19. Compared to the random selection results in Table 17, we notice that random selection is indeed a robust baseline at this data sparsity, as forgetting statistics fail to convincingly outperform. We however notice that interestingly for ticket the 10% most forgotten samples generalize better, while for the permuted ticket and dense network the least-forgotten samples perform better at 10% subset size.

Table 19: Validation accuracies for a 89.26% sparse ticket, a permuted version of the ticket and a dense **ResNet-18** model trained on different dataset subsets of **CIFAR-10**. Each model was achieved with $\mu = 0.0$, batch size 256.

| | Least forgotten subset | | Most forgotten subset | |
|---|---|---|---|---|
| Network | 1% | 10% | 1% | 10% |
| Ticket | 85.53 ± 0.12% | 87.28 ± 0.04% | 82.32 ± 0.91% | 91.40 ± 0.15% |
| Permuted | 47.12 ± 0.80% | 68.62 ± 0.25% | 31.51 ± 0.50% | 47.21 ± 0.26% |
| Dense | 46.54 ± 0.68% | 68.21 ± 0.52% | 31.87 ± 0.38% | 45.52 ± 0.74% |

## F Hyperparameter configurations

**General parameters.** Each training run contains of 200 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 parameters.** 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) 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. No LR scheduling is applied in this case.