MAX-AFFINE SPLINE INSIGHTS INTO DEEP NETWORK PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the importance of pruning in Deep Networks (DNs) and motivate it based on the current absence of data aware weight initialization. Current DN initializations, focusing primarily at maintaining first order statistics of the feature maps through depth, force practitioners to overparametrize a model in order to reach high performances. This overparametrization can then be pruned a posteriori, leading to a phenomenon known as "winning tickets". However, the pruning literature still relies on empirically investigations, lacking a theoretical understanding of (1) how pruning affects the decision boundary, (2) how to interpret pruning, (3) how to design principled pruning techniques, and (4) how to theoretically study pruning. To tackle those questions, we propose to employ recent advances in theoretical analysis of Continuous Piecewise Affine (CPA) DNs. From this viewpoint, we can study the DNs' input space partitioning and detect the early-bird (EB) phenomenon, guide practitioners by identifying when to stop the first training step, provide interpretability into current pruning techniques, and develop a principled pruning criteria towards efficient DN training. Finally, we conduct extensive experiments to shown the effectiveness of the proposed spline pruning criteria in terms of both layerwise and global pruning over state-of-the-art pruning methods. All the codes will be released publicly upon acceptance.

1 INTRODUCTION

Deep Networks (DNs) are powerful and versatile function approximators that have reached outstanding performances across various tasks, such as game playing (Silver et al., 2017), genomics (Zou et al., 2019), language processing (Esteva et al., 2019), and geophysics (Seydoux et al., 2020).

For decades, the main driving factors of DN performances haves been improvements in their *architectures* such as the form of the *nonlinear operators* (Glorot et al., 2011; Maas et al., 2013), or the arrangement of the succession of linear and nonlinear operators (LeCun et al., 1995; He et al., 2016; Zhang et al., 2018); producing more and more specialized architectures. With current DN performances and the need for practical deployment, a novel line of research aims to adapt those models to produce a simpler, energy efficient one by *pruning* a given architecture. Recent progresses (You et al., 2020; Molchanov et al., 2016) in that direction allow to obtain models requiring much less energy to produce a prediction which is crucial for various applications (Li et al., 2020).

While tremendous empirical progress has been made to perfect those techniques, there remains a lack of theoretical understanding on the impact of those techniques in the produced model prediction, and a lack of theoretical tools that would allow to derive novel techniques in a principled way; or analysis and compare existing ones. Such understandings are crucial for one to study the possible failure modes of those techniques, to better decide which to use based on a given application, or to design novel techniques possibly guided by some a priori known requirements from the task at hand.

We propose to shed some lights into the inner workings of those pruning techniques from a spline perspective leveraging recent advances in DN understandings and spline formulation (Montufar et al., 2014; Balestriero & Baraniuk, 2018). We will demonstrate how such formalism will allow to interpret current techniques by studying their impact on the DN input space partition, demonstrating *how* and *when* can pruning be used without sacrificing the final performances. Finally, we demonstrate how to derive a novel pruning scheme based on our gained understandings that reaches competitive performances. We summarize our contributions as follows:

[C1] We first motivate (DN) pruning by demonstrating how the general scheme of (i) overparametrization of a model, (ii) training of this "large" model, and (iii) pruning of this model to

produce a smaller one is able to provide practitioners with significant performance gains as opposed to using a small model throughout the training process, *in particular when good initialization of the model parameters is not known* (Sec. 3.1). We demonstrate that the above procedure indeed produces better performing models whenever usual initialization is employed but no longer holds if "smarter" initialization is employed open new avenues to produce performant small DNs (Sec. 3.2).

[C2] We propose to leverage the spline formulation of DNs to interpret current pruning techniques. We relate the pruning of DN units and/or layer weights to (i) the DN input space partition, (ii) the per-region affine parameters, and (iii) the decision boundary, from which we can study the difference of different pruning techniques, such as unit versus weight pruning (Sec. 4). In particular, we propose a partition based metric to quantify the evolution of the latter during training from which one can efficiently detect Early-Bird (EB) tickets (when an overparametrize DN has been trained enough and can be pruned), as opposed to current EB methods, ours does not require an external pruning technique and is easily interpretable (Sec. 4.2).

[C3] We leverage the findings from [C2] and theoretically motivate a novel pruning strategy based on pruning units that are redundant in term of the DN input space partition geometry (Sec. 4.3). We carefully detail this technique and perform series of experiments on various benchmarked datasets in which we observe novel state-of-the-art results (Sec. 5).

2 BACKGROUND AND RELATED WORK

Denote the input-output DN mapping as $f : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{X}$ the input space $\mathbb{R}^D$ and $\mathcal{Y}$ the prediction space such as $\mathbb{R}$ for univariate regression. An important property of DN lies in transforming an input $\boldsymbol{x}$ through a serie of L layers $f_\ell, \ell = 1, \ldots, L$ to form the final prediction $y = f(\boldsymbol{x})$. Each layer is a (nonlinear) operator f_ℓ mapping a $D^{\ell-1}$-dimensional *feature map* to a D^ℓ-dimensional one, where D^ℓ is the ℓth feature map's dimension. We collect each layer's parameters in θ_ℓ. For a fully-connected and activation function layer, θ_ℓ comprises the $D^\ell \times D^{\ell-1}$ dense matrix W^ℓ and the D^ℓ-dimensional bias vector $\boldsymbol{b}^\ell$. For convolution operators (LeCun et al., 1995), the dense matrix W^ℓ is replaced with the circulant block circulant matrix $\boldsymbol{C}^\ell$ for channel-wise convolution and summation.

Max-Affine Spline DNs. A key result of (Balestriero & Baraniuk, 2018) is the reformulation of current DN layers to spline operators and in particular the Max-Affine Spline Operator (MASO). With this formulation each layer's input-output mapping corresponds to

$$f^\ell(\boldsymbol{z}) = \max_{r=1,\ldots,R} \left(\boldsymbol{A}_r^\ell \boldsymbol{z} + \boldsymbol{v}_r^\ell \right), \tag{1}$$

with the maximum taken coordinate-wise. Whenever a DN can be expressed as a MASO, it becomes a continous piecewise affine mapping (Unser, 2018) with an inherent input space partition and per-region affine mappings. In fact, by the form of (1) it is clear that there exists an underlying layer input space partition based on the realization of the maximum operator; for a study of this partitioning, see (Montufar et al., 2014; Balestriero et al., 2019).

This partition Ω of the DN input space is composed of non-empty and non-intersecting regions ω that collectively represent the entire DN input space leading to the following DN input-output form

$$f(\boldsymbol{x}) = \sum_{\omega \in \Omega} \left(A_\omega \boldsymbol{x} + \boldsymbol{b}_\omega \right) \mathbb{1}_{\boldsymbol{x} \in \omega} \tag{2}$$

where the above parameters denote the ω-associated affine parameters. The fundamental property that we will leverage throughout this paper is as follows. *The internal weights of the DNs are paired with the DN partition Ω. As pruning impacts those weights directly, this will offer us new ways to study and extend DNs pruning.*

Network Pruning. Pruning is a widely used DN compression technique for reducing the number of activated units (Liu et al., 2019; LeCun et al., 1990). The common pruning scheme adopts a three-step routine: (i) training a large model with more parameters/units than the desired final DN, (ii) pruning this overly large trained DN, and (iii) fine tune the pruned model to adjust the remaining parameters and restore performance that was lost due to pruning. Those three steps can be iterated to get a highly-sparse network (Han et al., 2015). Within this routine, different pruning methods can be employed each with a specific pruning criteria, granularity, and scheduling (Liu et al., 2019; Blalock et al., 2020). Those techniques roughly fall into two categories: unstructured pruning (Han et al.,
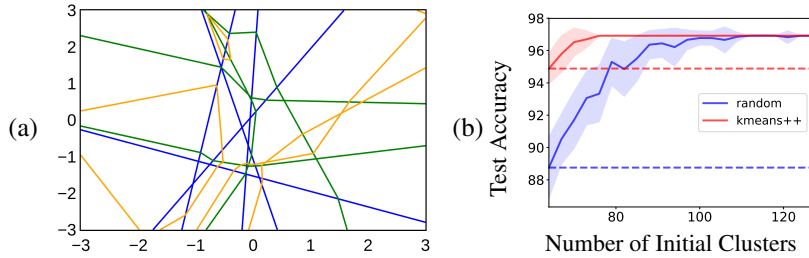
Figure 1: **(a):** DN input space partition for $L = 3, D^\ell = 5, \forall \ell$, the boundaries induced by layer 1,2,3 are depicted in blue, green and orange respectively. We can see how deeper layers successively subdivide the space to form the entire DN input space partition where in each region the DN behaves linearly. **(b):** K-means experiment on a toy mixture of 64 Gaussian in 2d (see Fig. 5). In all cases the number of final cluster is 64 but the number of starting clusters (x-axis) varies and pruning is applied during training to remove redundant centroids comparing random centroid initialization and kmeans++. With overparametrization, random initialization and pruning reaches same performance as kmeans++.

2015; Frankle & Carbin, 2019; Evci et al., 2019) and structured pruning (He et al., 2018; Liu et al., 2017; Chin et al., 2020). Regardless of pruning methods, the trade-offs lies between the amount of pruning performed on a model and the final accuracy. For various energy efficient applications, producing novel pruning techniques able to push this trade-off favorably is crucial.

Winning Tickets. (Frankle & Carbin, 2019) first hypothesised the existence of a sub-networks (pruned DNs), called winning tickets, that can produce comparable accuracies than their non-pruned counterpart. Later, (You et al., 2020) showed that those winning tickets can be identified in the early training stage of the un-pruned model. Such sub-networks are denoted as early-bird (EB) tickets.

Despite the above discoveries, the DN pruning literature lacks of a theoretical analysis that would both bring insights into (i) current pruning techniques and (ii) observed phenomenons such as EBs tickets, while providing new principled methods to design novel pruning techniques. We propose to approach this task by leveraging the spline theory (Balestriero & Baraniuk, 2018) for representing pruning and analyzing those techniques and their impact on DNs' decision boundaries.

3 WINNING TICKETS AND DEEP NETWORK INITIALIZATION

We propose in this section a simple example illustrating the concept of winning tickets and its link with the difficulty of optimizing a non overparametrized MASO DN.

3.1 THE INITIALIZATION DILEMMA AND THE IMPORTANCE OF OVERPARAMETRIZATION

The term *overparametrization* has been used extensively in the recent DN literature. Throughout this paper, we will refer to a model as being overparametrized when it is possible to solve the task at hand with same or better performance but with the model that has reduced capacity (and parameters). This can be done by reducing the number of units (and layers) in a DN, or reducing the number of centroids in K-means (MacQueen et al., 1967). In this section we propose to consider not only DNs but also more standard machine learning algorithms such as K-means with the following goal: demonstrate that *the chaining of (i) overparametrization (ii) training and (iii) pruning provides a powerful strategy when good initialization of the employed algorithm is not known.*

The importance of initialization is crucial even in more standard methods such as K-means. In fact, a rich branch of research has been focusing on this initialization problem alone for decades (Bradley & Fayyad, 1998; Kanungo et al., 2002; Hamerly & Elkan, 2002; Arthur & Vassilvitskii, 2006; Celebi et al., 2013). Instead, we now demonstrate how one can easily start with a random initialization but in a highly overparametrized model (for example employing much more clusters than desired in K-means) and then prune the model a posterior to end up with a model containing the (exact) number of parameters desired (such as the desired number of clusters). Let first consider the case of K-means in a toy setting where we know a priori the number of clusters and we generate them with spherical and identical covariances in order to fully fall into the K-means data modeling. We perform training over multiple runs by varying the number of original clusters that are used for training, and in all cases, once training is completed, pruning is done to remove clusters based on a simple strategy that simply removes closest centroids recursively until the true number of clusters is obtained. Hence, in all cases, the final trained models have the same number of clusters regardless of the number of clusters used during training. We report the clustering accuracy in Fig. 1. We

observe distinctively the ability of such a method to result in very accurate models whenever the number of starting clusters is greater than the true one. In fact, it should be clear that due to random initialization of the centroids, the more are used at initialization, the more likely it becomes that are least one centroid will be near each of the clusters of the data distribution.

Current DN initialization techniques focus on maintaining feature maps through depth that remain with bounded statistics to avoid vanishing of exploding gradient (Glorot & Bengio, 2010; Sutskever et al., 2013; Mishkin & Matas, 2015). However, incorporating data information into the DN weights initialization as is done in Kmeans with methods such as kmeans++ (Arthur & Vassilvitskii, 2006) remain to be developed for DNs. Hence, overparametrization plays a key role in compensating random initialization and allowing successful training, and a posteriori, once can remove the redundant parameters and obtain a final model with much better performances as the non-overparametrized and non-pruned counterpart. This is the key motivation of Early Bird tickets. Recalling Fig. 1 (a), it is clear that overparametrization increases the number of subdivision lines and the chance to well position some of them. In addition, overparametrization has also been used favorably as a way to help gradient descent based techniques facilitating optimization (Arpit & Bengio, 2019); lastly, overparametrized also positions the initial parameters close to good local minima (Allen-Zhu et al., 2019; Zou & Gu, 2019; Kawaguchi et al., 2019) reducing the amount of updated needed during training. We formalize those points in the remark below.

Remark 1 *Winning tickets are the result of employing overly parametrized DNs. Overly parametrized DNs are simpler to optimize and produce parameters closer to good local minima. Overly parametrized DNs are needed as current optimization techniques can not escape poor local minima, and smart DN initialization (near good local minima) is unknown.*

We further support the above point in the following section where we demonstrate how the absence of good initialization coupled with non optimal optimization problem impacts performances unless overparametrization is used, in which cased winning tickets naturally emerge.

3.2 Better Deep Network Initialization

We saw in the previous section that the concept of winning tickets emerges from the need to overparametrize which in turn emerges naturally from architecture search and cross-validation as overparametrizing greatly facilitates training and thus improves final results. We now show that if a better initialization of DNs existed, one would not have to resort to overparametrization in order to reach high performances and thus one would not resort

Table 1: Accuracies of layerwise (LW) pretraining, random initialization and EB.

Setting	Pruning ratio	Random init.	EB train	LW pretrain
VGG-16 CIFAR-10	30%	92.86	**93.30**	93.02
	50%	93.04	**93.49**	93.30
	70%	92.76	92.85	**93.25**
	90%	89.65	90.24	**90.29**

to Early Bird tickets and such pruning techniques as the DN would already be "near minimal".

We convey the above point with a carefully designed experiment. We consider three cases. First, the case of employ a small DN with random weights initialized from Kaiming initialization (He et al., 2015). Second, we consider the same small DN but with weights initialized based on unsupervised layerwise pretraining. We consider this as a data aware initialization, even though actual training occur, since we do not use any label information (Belilovsky et al., 2019). In both those cases, training is done on the classification task in the same manner. Third, we consider a large DN which is trained based on the EB method (training-pruning-training). The final three models all have the same architecture (but different weights based on their own training method). We report classification results in Table 1 and see that especially for very small final DNs (high pruning ratios) EB model outperforms a randomly initialized DN, but in turn a well initialized DN is able to outperform EB training. From this, we see that *the ability of pruning methods and in particular EB to produce better-performing small DNs that using a small DN directly only lies in our lack of good initialization for Deep Networks.* In fact, for high pruning ratios, layerwise pretraining even offers a more energy efficient method overall (including the pretraining phase) than EB training as given in Table 4 and Fig. 7 in the Appendix A.2. This should open the door and motivate further study of such scheme as a possible alternative solution to produce energy efficient models.

With the above motivations and with the lack of efficient training methods for small DNs, EB training remains crucial. We thus propose to study pruning in-depth and how to develop new ones from a spline perspective, providing a universal solution to design efficient DNs across tasks and datasets.
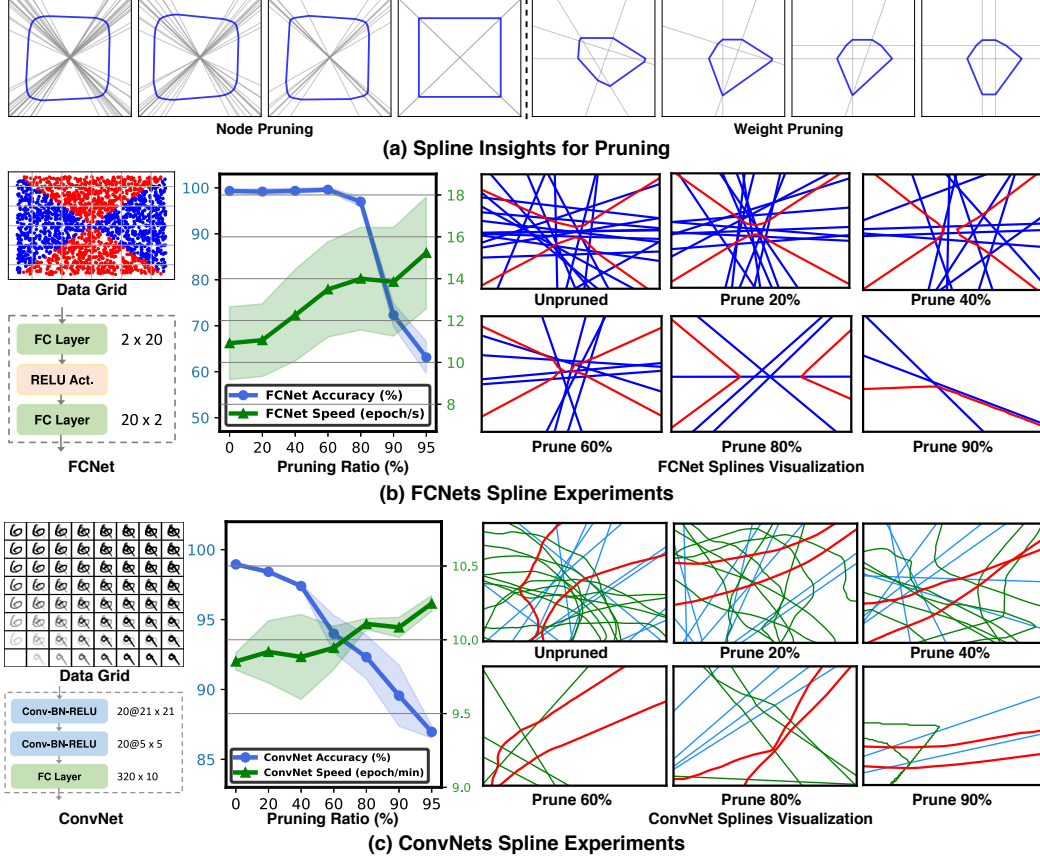
(a) Spline Insights for Pruning

(b) FCNets Spline Experiments

(c) ConvNets Spline Experiments

Figure 2: **(a) Difference between node and weight pruning**. The former removes entire subdivision lines while the latter simply quantise those partition lines to be colinear to the space axes. **(b) Toy classification task pruning.** Blue lines represent subdivisions in the first layer while red lines denote the last layer's decision boundary. We see that: 1) pruning indeed removes redundant subdivision lines so that the decision boundary remains X-shape until 80% nodes are pruned; 2) ideally one blue subdivision line would be sufficient to provide two turning points for the decision boundary, e.g., visualization at 80% sparsity, but the classification accuracy degrades a lot if further pruned. That aligns with the initialization dilemma for small DNs, i.e., blue lines are not well initialized and all lines remain hard for training. **(c) MNIST reproduction of (b).** To produce these visuals, we choose two images from different classes to obtain a 2-dimensional slice of the 764-dimensional input space (grid depicted on the left). We thus obtain a low-dimensional depiction of the subdivision lines that we depict in blue for the first layer, green for the second convolutional layer, and red for the decision boundary of 6 vs. 9 (based on the left grid). *The observation consistently shows that only parts of subdivision lines are useful for decision boundary; the goal of pruning is to remove those (redundant) subdivision lines.*

4 PRUNING WITH CONTINUOUS PIECEWISE AFFINE DEEP NETWORKS

Recall from Sec. 2 that a DN equipped with standard nonlinearities, such as (leaky-)ReLU/max-pooling, is a continuous piecewise affine spline with a partition of the DN input space. Such a connection provides us a new perspective to analyze *how the decision boundaries are gradually formulated* and *what does network compression methods (e.g., pruning and quantization) really mean from such a geometry perspective*. We propose to study those questions in the following sections.

4.1 INTERPRETING PRUNING FROM A SPLINE PERSPECTIVE

We first propose to leverage the DN input space partition to study the difference between node and weight pruning. In the former, units of different layers are removed while in the latter, entries of the W^{ℓ} matrix (or C for convolutions) are removed. We demonstrate in Fig. 2 (a) that *node pruning* removes entire subdivision lines while *weight pruning* (or quantization) can be thought as finer granular limitations on the slopes of subdivision lines, and will only remove the subdivision line when all entries of a specific row in W^{ℓ} are 0. From this, we can already identify the reason why pruned networks are less expressive than the overparametrized variants (Sharir & Shashua, 2018) as *pruned DNs input space partition is limited compared to their non-pruned counter parts*.

(a) Spline Trajectory for FCNets **(b) Spline Trajectory for ConvNets** **(c) Spline Early-Bird Tickets**
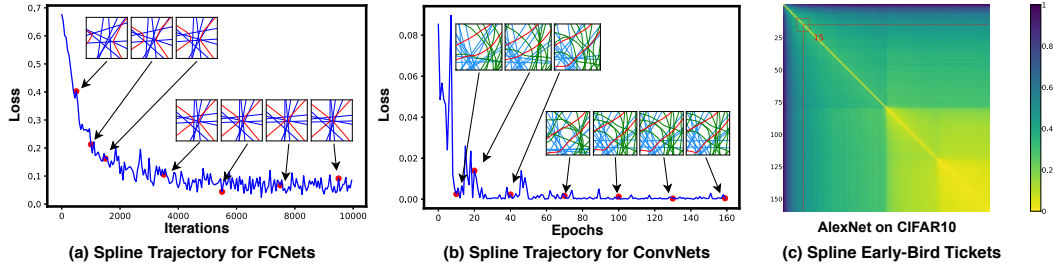
Figure 3: Spline trajectory during training and visualization of the Early-Bird (EB) Phenomenon, which can be leveraged to largely reduce the total training costs due to the less training of costly overparametrized networks. Those trajectories mainly adapt during early phase of training.

Despite this constraint on the DN input space partition that pruning imposes, classification performances do not necessarily reduce when pruning a model. In fact, the final decision boundary, while being tied with the DN input space partition, does not always depends on all the existing subdivision lines. That is, *pruning will not degrade performances as long as the imposed constraints on the DN input space partition are aligned with the needed decision boundary to solve the task.* We demonstrate and provide detailed visualization of the above in Fig. 2 (b) with a simple toy classification task which only requires a few subdivision lies to produce a decision boundary perfectly solving the task. Hence, as long as pruning leaves at least those few subdivision lines, the final performances will remain high. In fact, we observe that for this toy case, and with a two-layer FCNets (20 nodes per layer), applying pruning ratios ranging from from 20% ∼ 95% (i.e., prune 4 ∼ 19 nodes) does not prevent to solve the task as long as the pruned subdivision lines are not the necessary ones for the decision boundary. We also extend the above experiment to a high dimensional case with MNIST classification and a DN with two convolutional layers, 20 filters, and kernel sizes of 21 and 5, respectively in Fig. 2 (c). By adopting the same channel pruning method as in (Liu et al., 2017), we observe that most of the pruned nodes remove subdivision lines that were not crucial for the decision boundary and thus only has a small impact into the final classification performance.

4.2 SPLINE EARLY-BIRD TICKETS DETECTION

Early-Bird (EB) tickets (You et al., 2020) provides a method to draw winning ticket sub-networks from a large model very early during training (10% ∼ 20% of the total training epochs). The EB drawn is based on an a priori designed pruning strategy and in comparing how different (in term of which units/channels are removed) are the hypothetical pruned models through training steps; this methods outperforms SOTA methods (Frankle & Carbin, 2019; Liu et al., 2017). Its main limitation lies in the need to define a priori a pruning technique (itself depending on various hyper-parameters). Based on the spline formulation, we formulate a novel EB method that does not rely on an external technique and only considers the evolution of the DN input space partition during training.

Early-Bird in the Spline Trajectory. First, we demonstrate that there exists an EB phenomenon when viewing the DN input space partition, which should follow naturally as the DN weights and the DN input space partition are tied. We visualize DN partition's evolution at different training stages in Fig. 3 (a) and (b), under the same settings as Sec. 4.1. From this we clearly see that the partition quickly adapts to the task and data at hand, and then is only slightly refined through the remaining training epochs. This fast convergence comes as early as 2000-th iteration (w.r.t. 10000 iterations for FCNets) and 30-th epoch (w.r.t. 160 epochs for ConvNets). Additional, we observe that the contribution of the first layers in the input space partition becomes stable more rapidly than for deeper layers. We can thus leverage this early convergence to detect EB tickets by using a novel metric based on those subdivision lines to possibly draw better EB tickets than (You et al., 2020).

Quantitative Distance between Input Space Partitions. Besides the above qualitative visualization, we now propose a novel metric to perform EB ticket draws. First, recall that each region from the DN input space has an associated *binary code* based on which side of the subdivision trajectories the regions lies (Montufar et al., 2014; Balestriero & Baraniuk, 2018). Given a large collection of data points, we can assign to it the code of the region it lies in (found simply based on the sign of the per layer feature maps). As training happens and the partition adapts, the code associated to an input will vary. However, once training stabilises and regions remain almost the same, this code will remain constant for most of the inputs. In fact, one can easily show that in the infinite data sample regime covering the input space, DNs with same codes also have same input space partition, in turn synonym of same decision boundary geometry. Clearly, we can leverage this fact to detect when
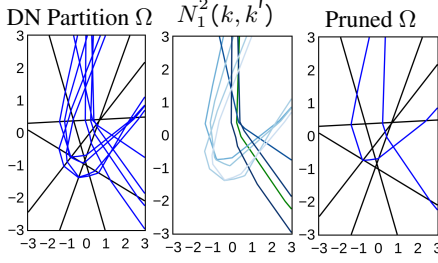
Figure 4: We depict on the **left** a small ($L = 2, D^1 = 5, D^2 = 8$) DN input space partition, layer 1 trajectories in black and layer 2 in blue. In the **middle** is the measure from 3 finding similar "partition trajectories" from layer 2 seen in the DN input space (comparing the green trajectory to the others with coloring based on the induce similarity from dark to light). Based on this measure, pruning can be done to remove the "grouped partition trajectoris" and obtain the pruned partition on the **right**.

learning stops adapting the input space partition significantly, and this can be done simply based on the hamming distance of those codes through consecutive training steps for the given samples.

We visualize the above hamming distance of the DN partition between consecutive epochs, when training AlexNet on CIFAR-10 (see more visualizations in Appendix B.1), shown as the spline distance matrix (160×160) in Fig. 3 (c), where (i, j)-th element represents the spline distance between networks from i-th and j-th epochs. The distances are normalized between 0 and 1, where a lower value (w.r.t. warmer temperature) indicates a smaller spline distance (and thus DNs with similar partitions). We consistently observe that such distance becomes small (i.e., < 0.15) in the first few epochs (visualization for other networks can be found in the Appendix) indicative of the EB phenomenun, but not capture in term of the DN input space partition. Therefore, we measure and record the spline distance between consecutive three epochs, and stop the training when all the recorded distances are smaller than a predefined threshold, denoted by the red block in Fig. 3 (c). Note that different from EB tickets drawn in (You et al., 2020), such spline EB tickets are invariant to the pruning ratios, and thus can be applied to the generic training process without any regularization.

4.3 SPLINE PRUNING POLICY

Recall from Sec. 2 that the layer input space partition is formed by a successive subdivision process involving each per layer input space partitioning. As we also studied in the previous section, for classification performances, not all the input space partition regions and boundaries are relevant since not all affect the final decision boundary. Knowing a priori which regions of the input space partition are helping in solving the task is extremely challenging since it requires knowledge of the decision boundary and of the input space partition, both being highly difficult to obtain for high dimensional spaces and large networks (Montufar et al., 2014; Balestriero et al., 2019; Hanin & Rolnick, 2019). We propose in this section an alternative method leveraging the spline formulation and providing a pruning strategy based on measuring how different units can be seen as redundant in term of their contribution to the entire DN input space partition.

When considering the layer input space partition we can identify "redundant" based on how this unit impacts the partition with respect to the other units. For example, if two units have same biases and colinear slope vector then one can effectively remove one of the two unit without altering the layer input space partition. While this is an extreme case, we will demonstrate that a "relaxed" version of the above, considering the angles between per-unit slope matrices and inter-bias distances allows to measure how redundant are two units into the forming of the DN input space partition. This pairwise measure if formally defined is as follows:

$$N_\rho^\ell(k, k') = \left(1 - \frac{|\langle [W^\ell]_{k,.}, [W^\ell]_{k',.} \rangle|}{\|[W^\ell]_{k,.}\|_2 \|[W^\ell]_{k',.}\|_2} \right) + \rho |[b^\ell]_k - [b^\ell]_{k'}|, \rho > 0, \tag{3}$$

where ρ is an hyper-parameter measuring the sensitivity of the difference in angle versus the biases. From this, finding the two units with the most similar contribution to the DN input space partitioning can be done via $\arg\min_{k,k' \neq k} N_\rho^\ell(k, k')$ where the obtained couple (k, k') encodes the two units which are the most redundant and thus can be pruned while altering the least the original DN input space partition. We illustrate this in Fig. 4 and provide the following formal result.

Proposition 1 *Given a layer and its input space partition, removing sequentially one of the two units k, k' for which $N_\rho^\ell(k, k') = 0$ leaves the layer input space partition unchanged.*

The above result is crucial as removing units that do not affect the layer input space is synonym of removing units that do not affect the entire DN input space partition, see (Balestriero et al., 2019) for details. In practice, units with small enough but nonzero $N_\rho^\ell(k, k')$ are also highly redundant and can be removed. We provide an example of this procedure in Fig. 4.

5 EXPERIMENT RESULTS

Here we evaluate our spline pruning method with the experiment settings added to Appendix C.1.

Table 2: Evaluating the proposed layerwise spline pruning over SOTA pruning methods.

Dataset	Pruning ratio	PreResNet-101				VGG-16			
		NS	Spline	EB Spline	Improv.	NS	Spline	EB Spline	Improv.
CIFAR-100	Unpruned	73.10	73.10	73.10	-	71.43	71.43	71.43	-
	10%	71.58	71.58	**73.14**	**+1.56**	71.6	71.78	**72.28**	**+0.68**
	30%	70.70	70.13	**72.11**	**+1.41**	70.32	71.15	**71.59**	**+1.27**
	50%	68.70	69.05	**70.88**	**+2.18**	66.1	69.92	**69.96**	**+3.86**
	70%	66.51	67.06	**68.41**	**+1.90**	61.16	63.13	**64.01**	**+2.85**

Table 3: Evaluating the proposed global spline pruning over SOTA pruning methods on ImageNet.

Models	Methods	Pruning ratio	Top-1 Acc. (%)	Top-1 Acc. Improv. (%)	Top-5 Acc. (%)	Top-5 Acc. Improv. (%)	Total Training FLOPs (P)	Total Training Energy (MJ)
ResNet-50	Unpruned	-	75.99	-	92.98	-	2839.96	280.72
	ThiNet	30%	72.04	-3.55	90.67	-2.31	4358.53	456.13
		50%	71.01	-4.58	90.02	-2.96	3850.03	431.73
	SFP	30%	74.61	-1.38	92.06	-0.92	4330.86	470.72
	EB Spline	30%	**75.08**	**-0.91**	**92.58**	**-0.40**	**2434.09**	**264.24**
		50%	**73.37**	**-2.62**	**91.53**	**-1.45**	**1636.02**	**197.09**

5.1 PROPOSED LAYERWISE SPLINE PRUNING OVER SOTA PRUNING METHODS

Recall the spline pruning policy by solving $\arg\min_{k,k'\neq k} N_\rho^\ell(k, k')$, we regard k as the index of channels and then are able to conduct channel pruning in a layerwise manner. Table 2 and Table 5 (Appendix C.2) shows the comparison between the spline pruning (w/ and w/o EB detection) and SOTA network slimming (NS) method (Liu et al., 2017) on CIFAR-100 and CIFAR-10, respectively. We can see that the spline pruning consistently outperform NS, achieving -0.08% ~ 3.86% accuracy improvements. This set of results verify our hypothesis that removing redundant splines incur little changes in decision boundary and thus provide a good a priori initialization for retraining.

5.2 PROPOSED GLOBAL SPLINE PRUNING OVER SOTA METHODS

We next extend the analysis to global pruning, where the mismatch of the filter dimension in different layers impedes the cosine similarity calculation. To solve this issue, we adopt PCA for reducing the feature dimensions to the same, before applying the spline pruning policy.

Spline Pruning over SOTA on CIFAR. Table 6 compares the retraining accuracy and total training FLOPS/energy of spline pruning with four SOTA pruning methods, whose detailed descriptions are in Appendix C.3. These results show that spline pruning consistently outperforms all competitors in terms of the accuracy and computational costs trade-offs. Specifically, compared with **the strongest competitor** among the four SOTA baselines, spline pruning achieves $0.8\times$ ~ $3.5\times$ training FLOPs reductions while offering comparable or even better (-0.67% ~ 0.69%) accuracies.

Spline Pruning over SOTA on ImageNet. We further investigate whether the spline pruning have consistent performance in a harder dataset, using ResNet-18/50 on ImageNet and benchmarking with ThiNet (Luo et al., 2017) and SFP (He et al., 2018). Specifically, spline pruning with EB detection achieves a reduced training FLOPs of 43.8% ~ 57.5% and a reduced training energy of 42.1% ~ 54.3% for ResNet-50, while leading to a top-1 accuracy improvement of 0.47% ~ 3.04% (a top-5 accuracy improvement of 0.52% ~ 1.91%). Results for ResNet-18 are in Appendix C.3.

The above experiments show the consistent superiority of global spline pruning. We also conduct ablation studies to measure the sensitivity of the hyperparameter ρ (see Equ. 3) in Appendix C.4.

6 CONCLUSIONS

We demonstrated the tight link between the presence of winning tickets and overparametrization in DNs, the latter being necessary for DNs to reach high performances with current (random) initializations; we demonstrated that this phenomenon is not unique to DNs but affect other machine learning methods such as K-means. This opens new avenues to produce small, energy efficient and performing DNs by developing working on initialization techniques. We leveraged the spline formulation of DNs to sharpen our understanding of different pruning policies, study the conditions in which pruning does not deteriorate performances and develop a novel and efficient pruning strategy extending EB tickets; extensive experiments demonstrated the superior performances (accuracy and energy efficiency) of the proposed method. The proposed spline viewpoint should open new avenues to theoretically study novel and existing pruning techniques as well as guide practitioners via the proposed visualization tools.

REFERENCES

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In Advances in neural information processing systems, pp. 6158–6169, 2019.

Devansh Arpit and Yoshua Bengio. The benefits of over-parameterization at initialization in deep relu networks. arXiv preprint arXiv:1901.03611, 2019.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

R. Balestriero and R. G. Baraniuk. A spline theory of deep networks. In Proc. Int. Conf. Mach. Learn., volume 80, pp. 374–383, Jul. 2018.

Randall Balestriero, Romain Cosentino, Behnaam Aazhang, and Richard Baraniuk. The geometry of deep networks: Power diagram subdivision. arXiv preprint arXiv:1905.08443, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In International conference on machine learning, pp. 583–593. PMLR, 2019.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In Third Conference on Machine Learning and Systems, 2020.

Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In ICML, volume 98, pp. 91–99. Citeseer, 1998.

M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert systems with applications, 40(1): 200–210, 2013.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. Nature medicine, 25(1):24–29, 2019.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. arXiv preprint arXiv:1911.11134, 2019.

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

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp. 315–323, 2011.

Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In Proceedings of the eleventh international conference on Information and knowledge management, pp. 600–607, 2002.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. arXiv preprint arXiv:1901.09021, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pp. 1026–1034, 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.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866, 2018.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. IEEE transactions on pattern analysis and machine intelligence, 24(7):881–892, 2002.

Kenji Kawaguchi, Jiaoyang Huang, and Leslie Pack Kaelbling. Effect of depth and width on local minima in deep learning. Neural computation, 31(7):1462–1498, 2019.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In Advances in neural information processing systems, pp. 598–605, 1990.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Chaojian Li, Tianlong Chen, Haoran You, Zhangyang Wang, and Yingyan Lin. Halo: Hardware-aware learning to optimize. In Proceedings of the European Conference on Computer Vision (ECCV), September 2020.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, pp. 2736–2744, 2017.

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

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE international conference on computer vision, pp. 5058–5066, 2017.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In Proc. icml, volume 30, pp. 3, 2013.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, pp. 281–297. Oakland, CA, USA, 1967.

Dmytro Mishkin and Jiri Matas. All you need is a good init. arXiv preprint arXiv:1511.06422, 2015.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440, 2016.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Advances in neural information processing systems, pp. 2924–2932, 2014.

Léonard Seydoux, Randall Balestriero, Piero Poli, Maarten De Hoop, Michel Campillo, and Richard Baraniuk. Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning. Nature communications, 11(1):1–12, 2020.

Or Sharir and Amnon Shashua. On the expressive power of overlapping architectures of deep learning. In International Conference on Learning Representations, 2018. URL `https://openreview.net/forum?id=HkNGsseC-`.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. nature, 550(7676):354–359, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In International conference on machine learning, pp. 1139–1147, 2013.

Michael Unser. A representer theorem for deep neural networks. arXiv preprint arXiv:1802.09210, 2018.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. Drawing early-bird tickets: Toward more efficient training of deep networks. In International Conference on Learning Representations, 2020. URL `https://openreview.net/forum?id=BJxsrgStvr`.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2472–2481, 2018.

Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. In Advances in Neural Information Processing Systems, pp. 2055–2064, 2019.

James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti. A primer on deep learning in genomics. Nature genetics, 51(1):12–18, 2019.

SUPPLEMENTARY MATERIAL

The following appendices support the main paper and are organized as follows:

- Sec. A provides more experiments to support Sec. 3.
    - Sec. A.1 extends the insights between overparametrization and winning tickets.
    - Sec. A.2 supplies comprehensive comparisons between layerwise pretrain and EB train.
- Sec. B provides more visualizations to support Sec. 4.
    - Sec. B.1 visualizes the Early-Bird phenomenon of additional networks.
- Sec. C provides extensive experiments to enrich Sec. 5.
    - Sec. C.1 describes the detailed expreiment settings.
    - Sec. C.2 supplies layerwise spline pruning results on CIFAR-10.
    - Sec. C.3 supplies global spline pruning results on CIFAR-10/100 and ImageNet.
    - Sec. C.4 supplies ablation studies to investigate the sensitivity of the hyperparameter ρ.

A   ADDITIONAL RESULTS ON INITIALIZATION AND PRUNING

A.1   WINNING TICKETS AND OVERPARAMETERIZATION

We can extend the overparametrization-pruning versus initialization insights from Sec. 3.1 into univariate DNs in a carefully designed dataset. Let consider a simple unidimensional sawtooth as displayed in Fig. 6 with P peaks in general (here $P = 2$). In the special case of a single hidden layer with ReLU activation, one must have at least $2P$ units to perfectly fit this function with weight configuration being $[W_1]_{1,k} = 1, [b_1]_k = -k$ with $k = 1, \ldots, D_1$ and $[W_2]_{1,1} = 1, [W_2]_{1,k} (-2)^{(k-1)\%2}, k = 2, \ldots, D_1$. Note that those weights are not unique (other ones can identically fit the function) and are given as an example. At initialization, if the DN has only $2P$ units,

Figure 5: Depiction of the dataset used for the K-means experiment with 64 centroids.

the probability that a random weight initialization will arrange the initial splines in a way to allow training from gradient based optimization is low. Increasing the width of the initial network will increase the probability that some of the units are advantageously initialized throughout the domain and aligned with the natural input space partitioning of the target function (different regions for different increasing or decreasing sides of the sawtooth). This is what is empirically illustrated in Fig. 6 (right) where one can see that even repeating multiple initialization of a DN without overparametrization does not allow to solve the task, while overparametrizing, training, and then pruning to keep only the correct number of units allow for better approximation.
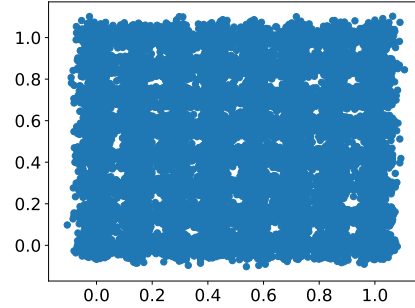
A.2   ADDITIONAL RESULTS FOR LAYERWISE PRETRAINING

We have shown that unsupervised layerwise pretraining provides a good initialization for small networks in the main paper. Here we supply more results and analysis to support that point and discuss other interesting trade-offs between overparametrization and layerwise pretraining.

Table 4 reports the classification results on VGG-16 and CIFAR-10/100 for three settings: random initialization, layerwise pretraining, and EB training, among which only EB training starts from overparametrization and then prune and retrain to recover the performance of the pruned network. We can see that the results consistently support the point in the main paper that, EB training (i.e., inherit weight from overparametrized models) outperform a randomly initialized small DN, while a
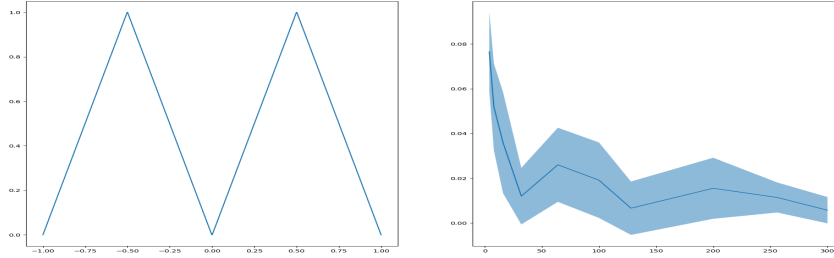
Figure 6: Depiction of a simple (toy) univariate regression task with target function being a sawtooth with two peaks (left). On the right is the ℓ_2 training error based on how overparametrized is the initial DN. In all cases, the final DNs have the same number of units (pruning ratios are adapted from the initial number of units).

Table 4: Comparing layerwise pretraining, random initialization, and Early Bird (EB) training.

Model	Pruning ratio	VGG-16		
		Random init	EB train	LW pretrain
CIFAR-10	30%	92.86	**93.30**	93.02
	50%	93.04	**93.49**	93.30
	70%	92.76	92.85	**93.25**
	90%	89.65	90.24	**90.29**
CIFAR-100	10%	71.70	**71.68**	71.06
	30%	71.88	**72.44**	72.31
	50%	70.64	**71.63**	70.63
	70%	66.14	66.30	**67.99**
	90%	48.04	49.42	**51.60**

well initialized small DN (i.e., after layerwise pretraining) performs good and even better than the EB training when applying high pruning ratios.

Fig. 7 further compares the required FLOPs of EB training and layerwise pretraining. We observer that 1) when the pruning ratio is low (i.e., < 50%), EB training consumes less computational FLOPs for getting good initialization of the pruned architecture, while leading to a comparable or even higher retraining accuracy; 2) when pruning ratio is higher, layerwise pretraining requires much less computational FLOPs as compared to training highly overparametrized dense networks. Such phenomenon opens a door for investigating below two questions, which we leave as further works.

- Is there a clear boundary/condition to show whether we should start from overparametrization or consider pretraining as a good initialization for samll DNs instead?
- How much overparametrization do we need to maintain better trade-offs between accuracy and efficiency, as compared to other initialization ways (e.g., layerwise pretraining)?
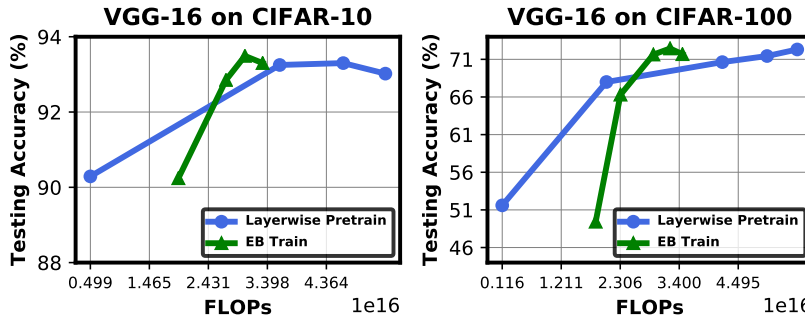

Figure 7: Accuracy vs. efficiency trade-offs of EB training and layerwise pretraining.
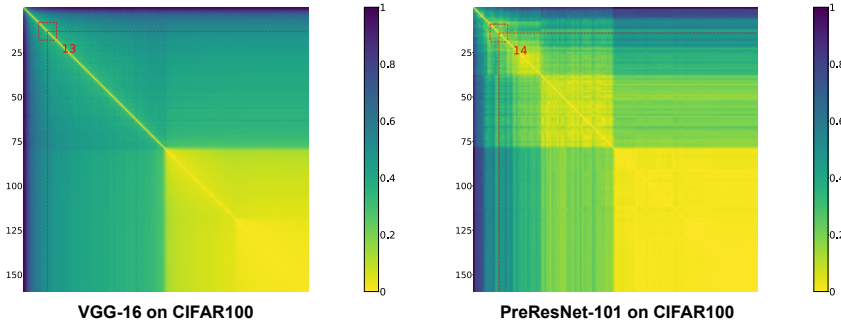
13

Figure 8: Spline Early-Bird tickets in VGG-16 and PreResNet-101.

B ADDITIONAL EARLY-BIRD VISUALIZATIONS

B.1 EARLY-BIRD VISUALIZATION FOR VGG-16 AND PRERESNET-101

We here supply more visualizations for the spline Early-Bird detection, which measures the hamming distances of the DN partition between consecutive epochs. Fig. 8 shows such visualizations for VGG-16 and PreResNet-101 networks evaluating on CIFAR-100 dataset. We see that both can identify spline EB tickets in the early training stages (i.e., 13/14-th epoch w.r.t. 160 epochs in total).

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 EXPERIMENTS SETTINGS

Models & Datasets: We consider four DNN models (PreResNet-101, VGG-16, and ResNet-18/50) on both CIFAR-10/100 and ImageNet datasets following the basic setting of (You et al., 2020).

Evaluation Metrics: We compare in terms of the retraining accuracy, total training FLOPs and energy costs, the latter of which are measured by training models on an edge GPU (NVIDIA JETSOn TX2), which considers both the computational and data movement costs.

Training Settings: For CIFAR-10/100 datasets, the training takes 160 epochs; the initial learning rate is set to 0.1 and is divided by 10 at the 80-th and 120-th epochs, respectively. For ImageNet, the training takes 90 epochs while learning rate drops at the 30-th and 60-th epochs instead. Batch size is set to 256 and a SGD solver is adopted with a momentum of 0.9 and a weight decay of 0.0001 for all experiments following the setting of (Liu et al., 2019). ρ in Equ. 3 is set to 0.05 for all cases.

C.2 ADDITIONAL RESULTS ON LAYERWISE SPLINE PRUNING

Table 5 supplies additional comparisons between the spline pruning (w/ and w/o EB detection) and baseline network slimming (NS) method (Liu et al., 2017), by evaluating VGG-16 and PreResNet-101 on CIFAR-10 dataset. We see that the spline pruning consistently outperform NS, achieving -0.08% ~ 1.16% accuracy improvements. This set of results verify our hypothesis that removing redundant subdivision lines incur little changes in decision boundary and thus provide a good priori initialization for retraining.

Table 5: Evaluating the proposed layrewise spline pruning over SOTA NS methods on CIFAR-10.

Dataset	Pruning ratio	PreResNet-101				VGG-16			
		NS	Spline	EB Spline	Spline Improv.	NS	Spline	EB Spline	Spline Improv.
CIFAR-10	Unpruned	93.66	93.66	93.66	-	92.71	92.71	92.71	-
	30%	93.48	**93.56**	93.07	**+0.08**	**93.29**	93.21	92.83	-0.08
	50%	92.52	**92.55**	92.37	**+0.03**	91.85	92.13	**92.23**	**+0.38**
	70%	91.27	91.33	**91.33**	**+0.06**	88.52	**89.68**	88.65	**+1.16**

Table 6: Global spline pruning over SOTA methods on CIFAR. Note that the "Spline Improv." denotes the improvement of spline pruning (w/ or w/o EB) as compared to the strongest baselines.

Setting	Methods	Retrain acc.			Energy cost (KJ)/FLOPs (P)		
		p=30%	p=50%	p=70%	p=30%	p=50%	p=70%
PreResNet-101 CIFAR-10	LT (one-shot)	93.7	93.21	**92.78**	6322/14.9	6322/14.9	6322/14.9
	SNIP	93.76	93.31	92.76	3161/7.40	3161/7.40	3161/7.40
	NS	93.83	93.42	92.49	5270/13.9	4641/12.7	4211/11.0
	ThiNet	93.39	93.07	91.42	3579/13.2	2656/10.6	1901/8.65
	Spline	**94.13**	**93.92**	92.06	4897/13.6	4382/12.1	3995/10.1
	EB Spline	93.67	93.18	92.32	**2322/6.00**	**1808/4.26**	**1421/2.74**
	Spline Improv.	**0.3**	**0.5**	-0.46	**1.4x/1.2x**	**1.5x/2.5x**	**1.4x/3.2x**
VGG16 CIFAR-10	LT (one-shot)	93.18	93.25	**93.28**	746.2/30.3	746.2/30.3	746.2/30.3
	SNIP	93.2	92.71	92.3	**373.1/15.1**	**373.1/15.1**	373.1/15.1
	NS	93.05	92.96	92.7	617.1/27.4	590.7/25.7	553.8/23.8
	ThiNet	92.82	91.92	90.4	631.5/26.4	383.9/19.0	380.1/16.6
	Spline	**93.62**	**93.46**	92.85	643.5/26.4	603.4/25.0	538.1/19.6
	EB Spline	93.28	93.05	91.96	476.1/19.4	436.1/15.5	**370.7/11.1**
	Spline Improv.	**0.42**	**0.21**	-0.43	0.8x/0.8x	0.9x/1.0x	**1.0x/1.4x**
PreResNet-101 CIFAR-100	LT (one-shot)	71.9	71.6	**69.95**	6095/14.9	6095/14.9	6095/14.9
	SNIP	72.34	71.63	70.01	3047/7.40	3047/7.40	3047/7.40
	NS	72.8	71.52	68.46	4851/13.7	4310/12.5	3993/10.3
	ThiNet	73.1	70.92	67.29	3603/13.2	2642/10.6	1893/8.65
	Spline	**73.79**	**72.04**	68.24	4980/12.6	4413/10.9	4008/9.36
	EB Spline	72.67	71.99	69.74	**2388/5.44**	**1821/3.84**	**1416/2.46**
	Spline Improv.	**0.69**	**0.44**	-0.27	**1.3x/1.4x**	**1.5x/2.8x**	**1.3x/3.5x**
		p=10%	p=30%	p=50%	p=10%	p=30%	p=50%
VGG16 CIFAR-100	LT (one-shot)	**72.62**	71.31	**70.96**	741.2/30.3	741.2/30.3	741.2/30.3
	SNIP	71.55	70.83	70.35	**370.6/15.1**	**370.6/15.1**	370.6/15.1
	NS	71.24	71.28	69.74	636.5/29.3	592.3/27.1	567.8/24.0
	ThiNet	70.83	69.57	67.22	632.2/27.4	568.5/22.6	381.4/19.0
	Spline	72.18	**71.54**	70.07	688.3/28.0	605.2/22.9	555.0/19.4
	EB Spline	72.07	71.46	70.29	512.2/19.9	429.1/15.3	**378.9/11.8**
	Spline Improv.	-0.44	**0.23**	-0.67	0.7x/0.8x	0.9x/1.0x	**1.0x/1.3x**

C.3 Additional results on Global spline pruning

Spline Pruning over SOTA on CIFAR. Table 6 compares the retraining accuracy and total training FLOPS/energy of spline pruning with four SOTA pruning methods, including two unstructured pruning baselines: the original lottery ticket (LT) training (Frankle & Carbin, 2019) and SNIP (Lee et al., 2019), and two structured pruning baselines: NS (Liu et al., 2017) and ThiNet (Luo et al., 2017). The results demonstrates that spline pruning again consistently outperforms all competitors in terms of the accuracy and energy costs trade-offs. Specifically, compared with the strongest competitor among the four SOTA baslines, spline pruning achieves 0.8 × ~ 3.5 × total training FLOPs reductions and 0.7 × ~ 1.5 × energy cost reductions while offering comparable or even better (-0.67% ~ 0.69%) accuracies. In particular, spline pruning consistently achieve 1.16 × ~ 3.16 × total training FLOPs reductions than all structured pruning baselines, while leading to comparable or better accuracies (-0.17% ~ 1.28%).

Spline Pruning over SOTA on ImageNet. We supply the comparison on ResNet-18 and ImageNet here, benchmarking with NS (Liu et al., 2017), SFP (He et al., 2018). As shown in Table 7, spline pruning with EB detection achieves a reduced training FLOPs of 44.7% ~ 61.7% and a reduced training energy of 39.5% ~ 54.2%, while leading to comparable top-1 accuracies (-0.24% ~ 0.71%) and top-5 accuracies (-0.16% ~ 0.21%).

C.4 Abalation studies of spline pruning

Recalling the only hyper-parameters ρ in the spline pruning policy (Equ. 3), which balances the difference in angle versus the biases. We here conduct ablation studies to measure the retraining accuracies under different values of ρ for investigating its sensitivity, shown as Fig. 9. Without

Table 7: Evaluating the proposed global spline pruning over SOTA pruning methods on ImageNet.

Models	Methods	Pruning ratio	Top-1 Acc. (%)	Top-1 Acc. Improv. (%)	Top-5 Acc. (%)	Top-5 Acc. Improv. (%)	Total Training FLOPs (P)	Total Training Energy (MJ)
ResNet-18	Unpruned	-	69.57	-	89.24	-	1259.13	98.14
	NS	10%	69.65	+0.08	89.20	-0.04	2424.86	193.51
		30%	67.85	-1.72	88.07	-1.17	2168.89	180.92
	SFP	30%	67.10	-2.47	87.78	-1.46	1991.94	158.14
	EB Spline	10%	69.41	-0.16	89.04	-0.20	1101.24	95.63
		30%	67.81	-1.76	87.99	-1.25	831.00	82.85
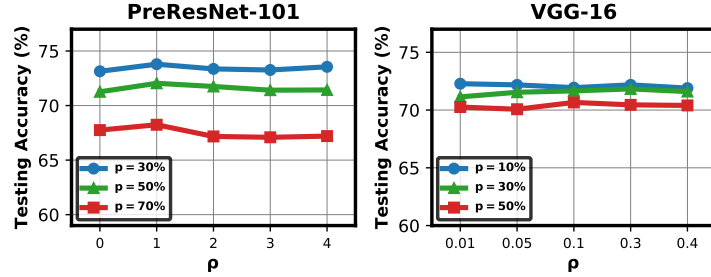
Figure 9: Abalation studies of the hyperparameter ρ in spline pruning.

loss of generality, we evaluate two commonly used models, VGG-16 and PreResNet-101, on the representative CIFAR-100 dataset. Results show that spline pruning consistently performs good for a wide range values of ρ staring from 0.01 to 0.4, which also generalizes to different pruning ratios (denoted by P). This set of experiments demonstrate the robustness of our spline pruning methods.