Max-Affine Spline Insights Into Deep Network Pruning

Anonymous authors
Paper under double-blind review

Abstract

State-of-the-art (SOTA) approaches to deep network (DN) training overparametrize the model and then prune a posteriori to obtain a "winning ticket" subnetwork that can achieve high accuracy. To date, the literature has remained largely empirical and hence provides little insights into how pruning affects a DN's decision boundary and no guidance regarding how to design a principled pruning technique. Using a recently developed spline interpretation of DNs, we develop new theory and visualization tools that provide insights into how pruning DN nodes affects its decision boundary. We discover that a DN's spline mappings exhibit an early-bird (EB) phenomenon whereby the spline's partition converges at early training stages, bridging the recently developed DN spline theory and lottery ticket hypothesis of DNs. We leverage this new insight to develop a principled and efficient pruning strategy whose goal is to prune isolate groups of nodes that have a redundant contribution in the forming of the spline partition. Extensive experiments on four networks and three datasets validate that our new spline-based DN pruning approach reduces training FLOPs by up to **3.5**× while achieving similar or even better accuracy than current state-of-the-art 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 board-game playing (Silver et al., 2017), genomics (Zou et al., 2019), and computer vision (Esteva et al., 2019). For decades, the main driving factor of DN performances has been progresses in their *architectures*, e.g. with the finding of novel nonlinear operators (Glorot et al., 2011; Maas et al., 2013), or by discovering novel arrangements of the succession of linear and nonlinear operators (LeCun et al., 1995; He et al., 2016; Zhang et al., 2018). With a tremendously increasing need for DNs' practical deployments, one line of research aims to produce a simpler, energy efficient DN by *pruning* a dense and overparametrized one, e.g. removing either weights, nodes, filters, layers, or any combination of these options from a DN architecture, leading to a much reduced computational cost (Frankle & Carbin, 2019b; Han et al., 2015; Chin et al., 2020; Liu et al., 2017). Recent progresses (You et al., 2020; Molchanov et al., 2016) in this direction allow to obtain models much more energy friendly while nearly maintaining the models' task accuracy (Li et al., 2020).

While tremendous empirical progress has been made regarding DN pruning, there remains a lack of explicit understanding of its impact on a DN's decision boundary plus a lack of theoretical tools for deriving pruning techniques in a principled way. Such understandings are crucial for one to study the possible failure modes of pruning techniques, to better decide which to use based on a given application, or to design pruning techniques possibly guided by some a priori knowledge about the given task and data. Our core motivation is to bridge the affine spline formulation of DNs and the recently developed empirical pruning techniques, e.g., lottery ticket hypothesis Frankle & Carbin (2019b); You et al. (2020); Evci et al. (2019); Su et al. (2020); Blalock et al. (2020).

In this paper, we shed new light on the inner workings of pruning techniques from a spline perspective, by leveraging recent advances in DN understandings and spline formulation (Montufar et al., 2014; Balestriero & Baraniuk, 2018). Specifically, current DNs are affine splines, that is the input-output mapping is affine in polytopal regions of the input space partition. From this viewpoint pruning acts upon a DN by removing/altering the partition boundaries as demonstrated in Fig. 1, and therefore pruning affects the decision

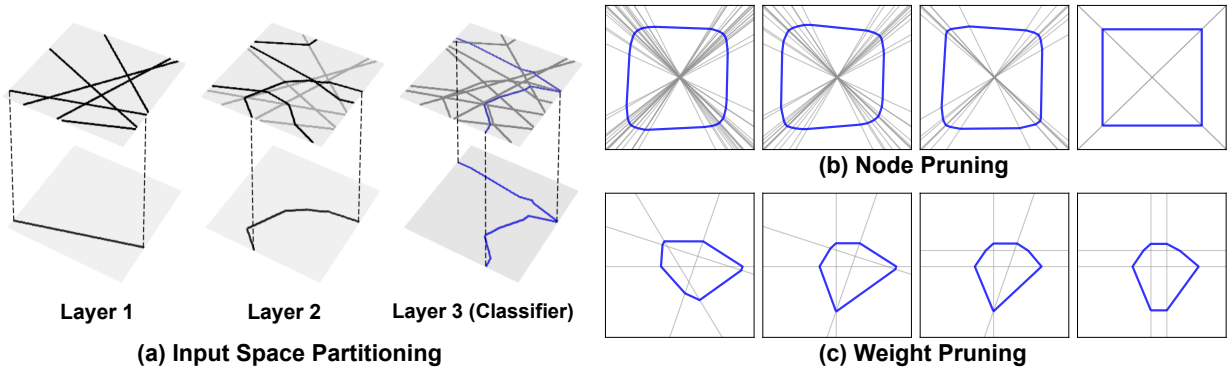

Figure 1: **(a) Input space partitioning** presents how deeper layers successively subdivide the space, where the newly introduced boundaries are in dark and previously built ones are in grey. **We see that:** *(i)* the turning point of splines in later layers are exactly located at previous ones, and *(ii)* splines in the final classification layer are exactly the decision boundary (denoted as blue lines). Additional examples are supplied in Appendix H; **(b) Node (structured) pruning** removes entire subdivision splines; **(c) Weight (unstructured) pruning** quantizes the partition splines to be colinear to the space axes. Both (b) and (c) lead to the less expressiveness of the final decision boundary.

boundary which is constrained to be linear within the regions of the DN partition. We will demonstrate how this viewpoint allows us to interpret current pruning techniques (e.g., lottery tickets hypothesis (Frankle & Carbin, 2019a)) 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 new pruning scheme based on our gained understandings that reaches competitive performances. In order to ease our development, we slightly abuse notations and refer to a DN as being overparametrized whenever it can be pruned while maintaining its performances and refer to a DN as being minimal whenever it can not be pruned without impacting its performances. We summarize our contributions as follows:

[C1] We discover and bridge the connection between spline theory and network pruning techniques. Specifically, we relate the pruning of DN nodes or weights to *(i)* the DN input space partition, *(ii)* the per-region affine parameters, and *(iii)* the decision boundary, providing the explicit interpretation of existing empirical pruning strategies at various granularity levels (either structured or unstructured).

[C2] We further extend these insights by proposing a partition-based metric to quantify the evolution of the partition boundaries during training, which allows us to efficiently detect early-bird (EB) tickets when an overparametrize DN has been trained enough and can be pruned; and as opposed to previous EB methods, ours detects EB tickets regardless of the employed pruning techniques or hyperparameters.

[C3] We leverage the new insights and the finding of [C2] to derive an efficient pruning strategy from first principles, which only focuses on DN nodes whose corresponding spline partition boundaries contribute to the final decision boundary. A series of experiments on various benchmarked models and datasets validate that our pruning method achieves 3.5× training FLOPs reduction and maintains similar or even better accuracies over state-of-the-art pruning techniques, while being principled and interpretable.

2 Background and related works

A DN transforms an input $\boldsymbol{x}$ through a composition of L layers $f^\ell, \ell = 1, \dots, L$ to form the final prediction: $f(\boldsymbol{x}) = (f^L \circ \cdots \circ f^1)(\boldsymbol{x})$. Each layer is a (nonlinear) mapping taking as input a $D^{\ell-1}$-dimensional feature map and producing a D^ℓ-dimensional one, where D^ℓ is the ℓth feature map's dimension[1]; each layer's parameters are collected in θ_ℓ. For a fully-connected and activation function layer, θ_ℓ comprises the $D^\ell \times D^{\ell-1}$ dense matrix $\boldsymbol{W}^\ell$ and the D^ℓ-dimensional bias vector $\boldsymbol{b}^\ell$. For convolution operators (LeCun et al., 1995), the dense matrix $\boldsymbol{W}^\ell$ is replaced with a circulant block circulant matrix $\boldsymbol{C}^\ell$ for channel-wise convolutions and summations.

[1] we consider matrices and tensors as flattened vectors for clarity

Max-affine spline DNs. A key result of (Montufar et al., 2014; Balestriero & Baraniuk, 2018) is the reformulation of current DN layers with (leaky-)ReLu/Linear/Abs. Value activation functions as spline operators and in particular as Max-Affine Spline Operator (MASO). With this formulation each layer's input-output mapping can be written as

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

with the maximum taken coordinate-wise. By the form of Eq. (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 (Balestriero et al., 2019). Jointly, the layers combine their input space partition to form the DN input space partition Ω. A few work have focused on studying the relation between the number of regions and the DN architecture (Hanin & Rolnick, 2019b), the analytical form of the partition (Balestriero et al., 2019), the upper bound in the number of regions (Montufar et al., 2014; Montúfar et al., 2021). The fundamental property that we will leverage throughout this paper is that *the internal weights of the layers are paired with their input space partition. As pruning impacts those weights and/or nodes directly, this will offer us new ways to study pruning in DNs' input space.* More background on DNs, their spline formulation, and its difference with our work can be found in Appendix F.

Network pruning. Pruning is a widely used DN compression technique reducing the number of activated nodes (Liu et al., 2019c; LeCun et al., 1990) in a given model. The common pruning scheme adopts a three-step routine: *(i)* training a large model with more parameters than the desired final DN, *(ii)* pruning this overly large trained DN, and *(iii)* fine-tuning the pruned model to adjust the remaining parameters and restore as best as possible the performance lost during the pruning step. 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., 2019c; Blalock et al., 2020). Those techniques roughly fall into two categories: unstructured pruning (Han et al., 2015; Frankle & Carbin, 2019b) and structured pruning (He et al., 2018; Liu et al., 2017; Chin et al., 2020). Regardless of pruning methods, the trade-offs lie between the amount of pruning performed on a model and the final accuracy. For various energy efficient applications, novel pruning techniques have been able to push this trade-off favorably. The most recent theoretical works on DN pruning relies on studying the existence of Winning Tickets. (Frankle & Carbin, 2019b) first hypothesized the existence of sub-networks (pruned DNs), called winning tickets, that can produce comparable performances to their non-pruned counterpart. Later, (You et al., 2020) showed that those winning tickets could be identified in the early training stage of the unpruned model. Such sub-networks are denoted as early-bird (EB) tickets.

Despite the above discoveries, the DN pruning literature lacks an explicit understanding and visualization via theoretical analysis that would bring insights into *(i)* current pruning techniques and *(ii)* observed phenomenons such as EB tickets, while leading to principled pruning techniques. We propose to approach this task by leveraging the spline viewpoint of DNs to provide novel interpretations of existing pruning techniques, study the conditions to their success and when should they be avoided, and finally, how to derive novel pruning strategies from first principles.

3 Deep Networks Partition, Decision Boundary and Pruning Work Hand-In-Hand

In this section, we first introduce our novel spline interpretation of pruning at various granularity levels. Then we extend such insights by detecting spline EB tickets through a pruning invariant partition-based metric. Finally, we leverage the new insights to derive an principle and efficient pruning strategy.

3.1 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, representing structured and unstructured pruning, respectively. In the former, nodes of different layers are removed, while in the latter, entries of the $\boldsymbol{W}^{\ell}$ matrices (or $\boldsymbol{C}$ for convolutions) are removed. We demonstrate in Fig. 1 (b) and (c) that *node pruning* removes entire subdivision splines while *weight pruning* (or quantization) can be thought as finer granular limitations on the slopes of subdivision splines, and will only remove the subdivision lines when all entries of a specific row in $\boldsymbol{W}^{\ell}$ are pruned, in which

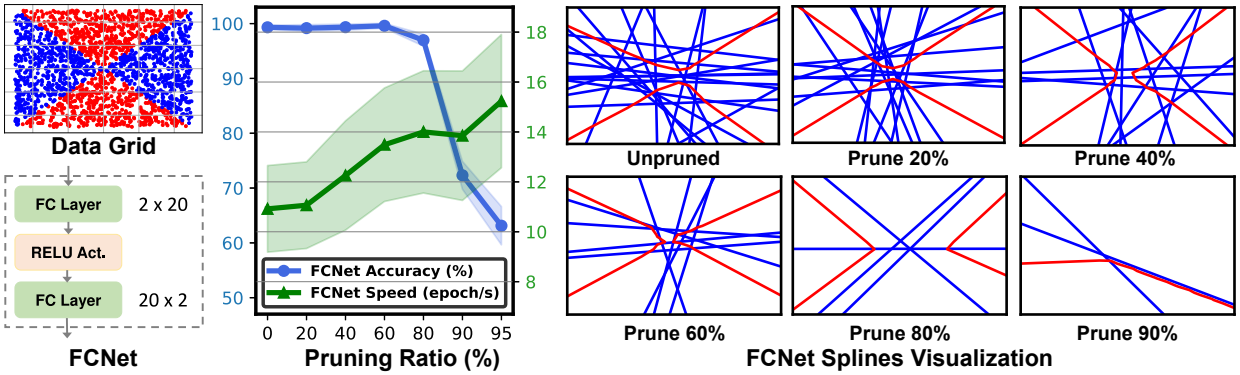

Figure 2: **Classification task pruning using FCNets**, where the blue lines represent subdivisions in the first layer and the red lines denote the last layer's decision boundary. **We see that:** *(i)* pruning indeed removes redundant subdivision lines so that the decision boundary remains an X-shape until 80% nodes are pruned; and *(ii)* ideally, one blue subdivision line would be sufficient to provide two turning points for the decision boundary, e.g., visualization at 80% sparsity.

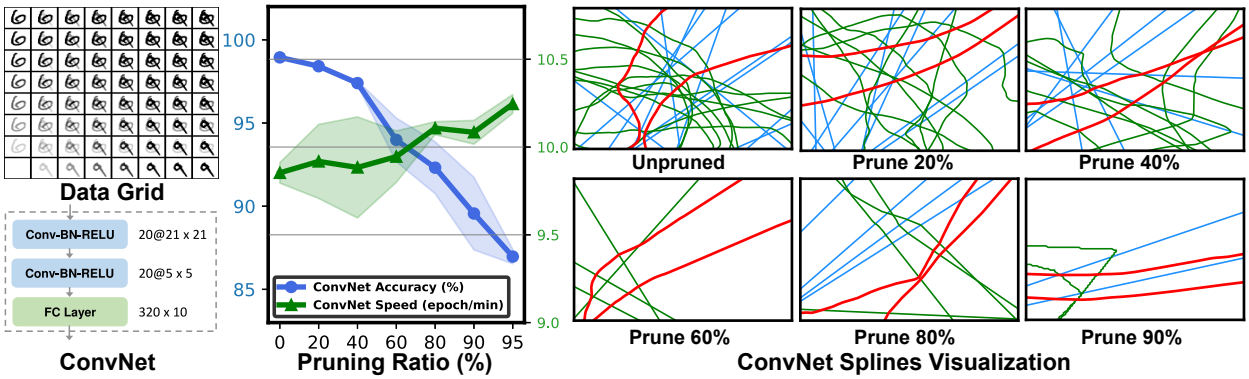

Figure 3: **Classification task pruning using ConvNets**, where to produce these visuals, we choose two images from different classes to obtain a 2-dimensional slice of the 764-dimensional input space (grid depicted on the left). We thus obtain a low-dimensional depiction of the subdivision splines that 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). We consistently find that only a fraction of splines are necessary to provide the turning points of final decision boundary.

case node and weight pruning become identical. From this perspective, 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 and final decision boundary shape are limited compared to their unpruned counterparts. Thus this limitation occurs first on the DN partition which in turn constrains the regions in the space where the decision boundary can be nonlinear as we recall that the DN decision boundary must be linear within the regions of the DN input space partition.

Despite the constraints that pruning imposes on the DN input space partition, classification performances do not necessarily reduce when pruning is employed. In fact, the final decision boundary, while being tied with the DN input space partition, does not always depend on all the existing subdivision lines. That is, *pruning will not degrade performances as long as the needed decision boundary geometry does not rely on the partition regions that are being affected by pruning.* We demonstrate and provide explicit visualization of the above in Fig. 2 and Fig. 3 with fully-connected networks (FCNets) and convolutional networks (ConvNets), respectively. **The observation consistently shows that only parts of subdivision splines are useful for decision boundary; and the goal of pruning is to remove those (redundant) subdivision splines and find winning tickets.** For example, we observe that for a two-layer FCNet (20 nodes per layer), applying pruning ratios ranging from 20% ~ 95% (i.e., prune 4 ~ 19 nodes) does not prevent solving

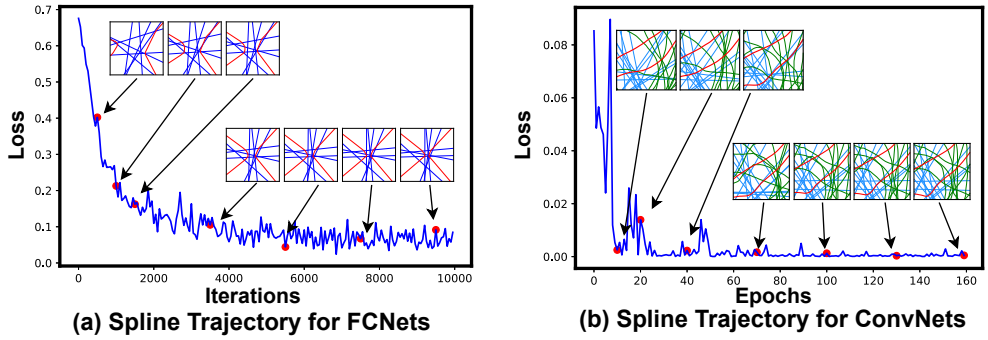

Figure 4: Visualization of spline trajectories, which mainly adapt during early phase of training demonstrating the lottery ticket hypothesis for DN partitions.

the task as long as the remaining subdivision lines are positioned to allow the decision boundary geometry to remain intact. 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. 3. By adopting the same channel pruning method as in (Liu et al., 2017), we see that most of the pruned nodes remove subdivision lines that were not crucial for the decision boundary and thus only have a small impact on the final classification performance. Hence, as long as pruning leaves at least those few subdivision lines, the final performances will remain high. Apart from the empirical study, we also provide some analysis about the connection between pruning (e.g, lottery ticket hypothesis (LTH)) and spline theory in Appendix G.

3.2 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 number of training epochs). The EB drawn is based on an a priori designed pruning strategy and hyperparameters and compares how different (in terms of which nodes/channels are removed) are the hypothetical pruned models through the training steps; this method outperforms SOTA methods (Frankle & Carbin, 2019b; Liu et al., 2017). The main limitation of EB lies in the need to define a priori a pruning technique (itself depending on various hyperparameters). 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. 4 (a) and (b), under the same settings as Sec. 3.1. 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 the 2000-th iteration (w.r.t. 10000 iterations for FCNets) and the 30-th epoch (w.r.t. 160 epochs for ConvNets). Additionally, 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 with **a novel metric based on those subdivision lines** to draw more unified EB tickets than (You et al., 2020). Moreover, EB tickets have been found to be *universal* under different optimization and initialization methods, of which the experiments are provided in Appendix D and E.

Quantitative distance between input space partitions. To draw EB tickets based on the evolution of DN input space partitions, we first need to provide a metric that conveys such information. 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 lie (Montufar et al., 2014; Balestriero & Baraniuk, 2018). Given a large collection of data points, we can assign each datum the code of the region it lies in (found simply based on the sign of the per-layer feature maps). As training occurs and the partition adapts, the code associated with an input will vary. However, once training stabilizes and regions do not change anymore, this code will remain the same. In fact, one can easily show that in the infinite data sample regime covering the entire input space, DNs with the same codes also have the same input space partition, in turn the same decision boundary geometry. The proposed metric is thus the hamming distance between the codes of each datum observed at two consecutive training steps.

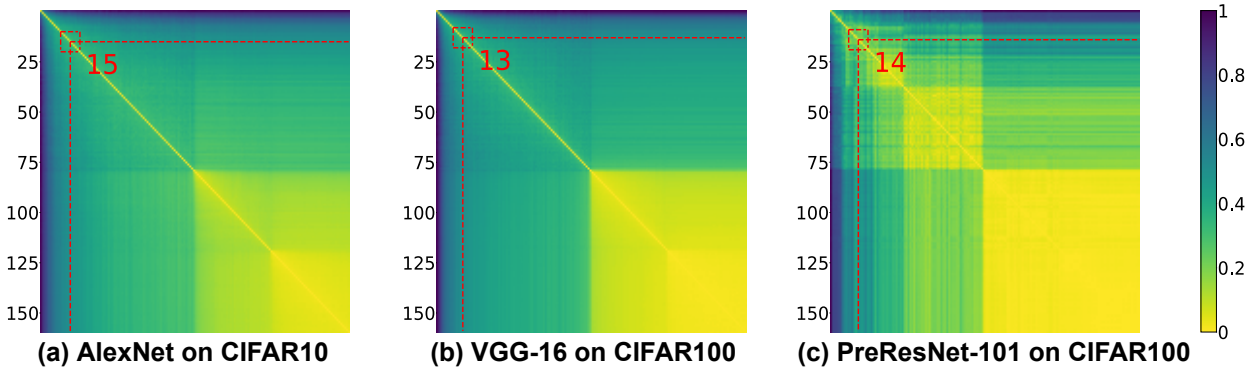

Figure 5: Visualization of the early-bird (EB) phenomenon, which can be leveraged to largely reduce the training costs due to the less training of costly overparametrized DNs.

We visualize the above hamming distance of the DN partition between 160 consecutive epochs, when training AlexNet on CIFAR-10 (shown as the spline distance matrix (160×160) in Fig. 5, where the (i, j)-th element represents the spline distance between networks from the 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 under different models and datasets settings. indicating the EB phenomenon, but now captured in terms of the DN input space partition. To obtain an active EB drawing strategy from that, we measure and record the spline distance between three consecutive epochs, and stop the training when the two associated distances are smaller than a predefined threshold of 0.15, denoted by the red block in Fig. 5. The detailed algorithm is provided in Appendix A. We conclude by emphasizing that as opposed to the usual EB tickets drawn in (You et al., 2020), our formulation provide a more **interpretable** scheme that is **invariant** to the pruning strategy as well as its hyperparameters (e.g., the pruning ratio). Hence, our formulation allows for a **unified** solution that does not require to be adapted based on the pruning technique that users experiment with.

3.3 Spline pruning policy

We now propose to derive from first principles novel pruning strategies of DNs based on the spline viewpoint insights. Recall from Sec. 2 that the layer input space partition is formed by a successive subdivision process involving each per-layer input space partition. 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 desired 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, 2019a). What is simpler to obtain, however, is how redundant are some of the layer weights/units in terms of the forming of the DN partition relative to other units/weights. From that, it will become easy to prune the redundant units/weights as their impact on the forming of the decision boundary is already carried by another unit/weight.

When considering the layer input space partition, we can **identify "redundant" units** based on how each unit impacts the partition with respect to other units. For example, if two units have biases and slope vectors proportional to each other, then one can effectively remove one of the two units without altering the layer input space partition. While this is a pathological case, we will demonstrate that the angles between per-unit slope matrices and inter-bias distances measure such a redundancy. We first introduce our pairwise redundancy measure as the following equation:

$$N^{\ell}_{\rho}(k, k') = \left(1 - \frac{|\langle [W^{\ell}]_{k,.}, [W^{\ell}]_{k',.} \rangle|}{\|[W^{\ell}]_{k,.}\|_2 \|[W^{\ell}]_{k',.}\|_2}\right) + \rho |[b^{\ell}]_k - [b^{\ell}]_{k'}|, \rho > 0, \tag{2}$$

where W_k refers to the corresponding slope matrices of the k-th unit, ρ is an hyper-parameter measuring the sensitivity of the difference in angle versus the biases. In the case of a convolutional layer, W_k is the flattened filter of shape $(\text{channels_in}, \text{height}, \text{width})$. 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. In turn, one of those two units can be pruned such that the impact of pruning onto the DN input space partition is minimized.

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

The above result exploits (Balestriero et al., 2019). In short, if $N_\rho^\ell(k,k')$ for any positive ρ, then the layer-partition boundaries (of layer ℓ) of units k,k' perfectly overlap. The DN partition boundaries correspond to the layer-partition boundaries backpropagated through the earlier layers to reach the data space. But this process is a continuous operator i.e., the layer-partition boundaries that overlap will produce DN partition boundaries that overlap. 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. 6, where a small DN input space partition (layer 1 trajectories in black and layer 2 in blue) is depicted. In the middle we visualize the measurements from Eq. (2) that trying to find 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.

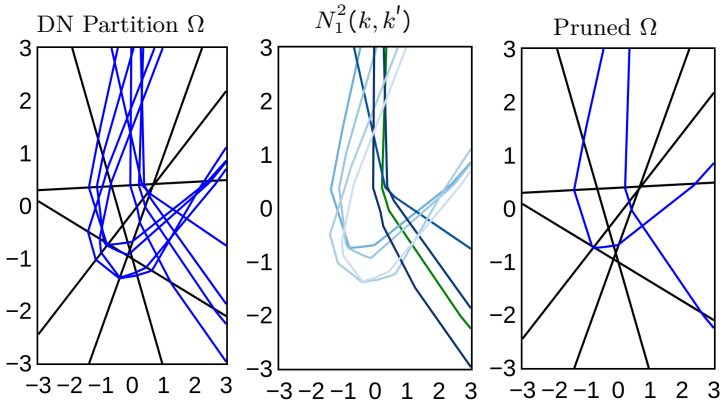

Figure 6: **Left:** a small $(L = 2, D^1 = 5, D^2 = 8)$ DN input space partition. **Middle:** the pruning criteria as in Eq. (2). **Right:** the pruned input space partition based on the criteria.

Efficiency of the spline pruning. Note that instead of using parameter-wise pruning, we perform channel-wise pruning of which the pruning overhead is negligible when evaluating across four models and three datasets. For example, ResNet-50 has about 20k channels and needs 100G FLOPs to prune, which is only 1/30,000,000 of the training costs (3000P FLOPs).

4 Experiment results

In this section, we first describe our experiment settings, and then benchmark the proposed spline pruning method over ten SOTA pruning baselines in the context of layerwise pruning and gloabl pruning, respectively. Finally, we present ablation studies in terms of the only hyper-parameter ρ.

4.1 Experiment settings

Models, datasets, baslines, and metrics: Models & Datasets: We consider four DNN models (PreResNet-101, VGG-16, and ResNet-18/50) on both the CIFAR-10/100 and ImageNet datasets following the basic setting of (You et al., 2020). Baselines: We evaluate the proposed spline pruning methods over ten SOTA training and pruning baselines, including network slimming (NS) (Liu et al., 2017), lottery tickets (LT) (Frankle & Carbin, 2019a), SNIP (Lee et al., 2019), ThiNet (Luo et al., 2017), SFP (He et al., 2018), LeGR (Chin et al., 2020), GAL-0.5 (Lin et al., 2019), GDP (Lin et al.), C-SGC-50 (Ding et al., 2019), and meta pruning (Liu et al., 2019b). Metrics: We evaluate in terms of the retraining accuracy, total training FLOPs, and real-device energy cost, the latter of which are measured by training the models on an edge GPU (NVIDIA JETSON TX2) (NVIDIA Inc.), which considers both the computational and data movement costs.

Training settings: For the CIFAR-10/100 datasets, the training takes a total of 160 epochs; and the initial learning rate is set to 0.1 and is divided by 10 at the 80-th and 120-th epochs, respectively. For the ImageNet dataset, the training takes a total of 90 epochs while the learning rate drops at the 30-th and 60-th epochs, respectively. In all the experiments, the batch size is set to 256, and an SGD solver is adopted with a momentum of 0.9 and a weight decay of 0.0001, following the setting of (Liu et al., 2019c). Additionally,

Table 1: Evaluating the proposed layerwise spline pruning over SOTA pruning methods on CIFAR-100, where all accuracies are averaged over five runs and "Improv." denotes the improvements of our layerwise spline pruning over the network slimming (NS) method.

Dataset	**Pruning ratio**	**PreResNet-101**				**VGG-16**			
		NS	Spline	EB Spline	**Improv.**	NS	Spline	EB 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**
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 2: Evaluating our global spline pruning method over SOTA methods on CIFAR-10/100 datasets. Note that the "Spline Improv." denotes the improvement of our spline pruning (w/ or w/o EB) **as compared to the strongest baselines**. All accuracies are averaged over five runs.

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	631.5/22.6	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**

ρ in Equ. 2 is set to 0.05 for all experiments except for the ablation studies. All experiments are run in a server with ten NVIDIA 2080 Ti GPUs.

4.2 Layerwise spline pruning

Recall that the spline pruning policy is done by solving $\arg\min_{k,k'\neq k} N_\rho^\ell(k,k')$. By regard k as the index of channels for convolutional layers, we are able to conduct channel pruning in a layerwise manner. Table 1 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-10/100 datasets. We can see that the spline pruning consistently outperforms NS, achieving -0.08% ~ 3.86% accuracy improvements. This set of results verifies our hypothesis that removing redundant splines incurs little changes in decision boundary and thus provides a good a priori initialization for retraining.

4.3 Global spline pruning

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, in practice, we adopt PCA (Scholz et al., 2008) for reducing the feature dimensions to the same before applying the spline pruning. PCA is well motivated as we aim to maintain as much information about the filters while reducing their dimensionality. However, we also consider factor analysis (FA) dimension reduction to demonstrate that spline pruning is not sensitive to the adopted dimension reduction methods as long as it can be used to measure the correlation between two units, and these FA experiments are provided in Appendix C.

Spline pruning over SOTA on CIFAR. Table 2 compares the retraining accuracy, the total training FLOPs, and the total training energy of our spline pruning methods with four SOTA pruning baselines, including two unstructured pruning baselines (i.e., the original lottery ticket (LT) training (Frankle & Carbin, 2019b) and SNIP (Lee et al., 2019)) and two structured pruning baselines (i.e., NS (Liu et al., 2017) and ThiNet (Luo et al., 2017)). The results demonstrate that our spline pruning again consistently outperforms all the competitors in terms of the achieved accuracy and training efficiency trade-offs. Specifically, compared with the strongest competitor among the four SOTA baselines, spline pruning achieves **0.8** × ~ **3.5** × 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 achieves **1.16** × ~ **3.16** × training FLOPs reductions than all the structured pruning baselines, while leading to comparable or better accuracies (-0.17% ~ 1.28%). More comparisons with baselines of pruning at initialization can be found in Appendix B.

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 eight SOTA pruning methods including ThiNet, NS, SFP, LeGR, GAL-0.5, GDP, C-SGD-50, and Meta Pruning. Specifically, spline pruning with EB detection (EB Spline) 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 -0.12% ~ 3.04% (a top-5 accuracy improvement of 0.18% ~ 1.91%). Consistently, EB Spline achieves a reduced training FLOPs of **44.7%** ~ **61.7%** and a reduced training energy of **39.5%** ~ **54.2%** for ResNet-18, while leading to comparable top-1 accuracies (-0.24% ~ 0.71%) and top-5 accuracies (-0.16% ~ 0.21%).

4.4 Ablation studies of the spline pruning method

Recalling that the only hyper-parameters ρ in our spline pruning method (see Equ. 2 of the main content), which balances the difference between the angles versus the biases. Here we conduct ablation studies to measure the retraining accuracies under different values of ρ for investigating its sensitivity, as shown in Fig. 7. Without 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 well for a wide range of ρ values ranging from 0.01 to 0.4,

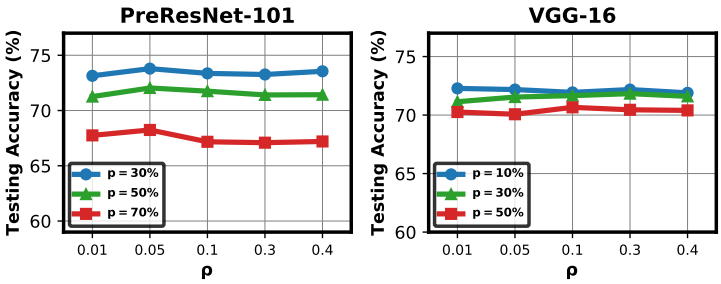

Figure 7: Abalation studies of the hyperparameter ρ in our spline pruning on two models, VGG-16 and PreResNet-101.

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-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**
ResNet-50	Unpruned	-	75.99	-	92.98	-	2839.96	280.72
	ThiNet	30%	72.04	-3.95	90.67	-2.31	4358.53	456.13
		50%	71.01	-4.98	90.02	-2.96	3850.03	431.73
	SFP	30%	74.61	-1.38	92.06	-0.92	4330.86	470.72
	LeGR	50%	75.3	-0.69	92.4	-0.58	4174.74	412.66
	GAL-0.5	40%	72.0	-3.99	91.8	-1.18	4458.74	440.73
	GDP	40%	72.6	-3.39	91.1	-1.88	4487.14	443.54
	C-SGD-50	50%	74.5	-1.49	92.1	-0.88	4117.94	407.04
	Meta Pruning	50%	73.4	-2.59	-	-	3532.63	349.12
	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**

which also generalizes to different pruning ratios (denoted by $\mathbf{p}$). This set of experiments demonstrate the robustness of our spline pruning methods.

5 Conclusions

We discover and bridge the connection between spline theory and network pruning techniques, providing explicit visualization and new insights into how pruning DN nodes affects the decision boundary, which well explains the presence of winning tickets and the importance of obtaining good initialization staring from overparametrization. Moreover, we extend these insights by proposing a pruning invariant metric to quantify the evolution of splines during training and detect the unified spline EB tickets. Finally, 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 more principled pruning strategy extending spline EB tickets; and extensive experiments demonstrated the superior performances (accuracy and energy efficiency) of the proposed method. The proposed spline viewpoint opens 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.

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.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4943–4953, 2019.

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, 2019a. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

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

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, 2019a.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 2596–2604. PMLR, 09–15 Jun 2019b. URL `https://proceedings.mlr.press/v97/hanin19a.html`.

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

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.

Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.

Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2790–2799, 2019.

Shengchao Liu, Dimitris Papailiopoulos, and Dimitris Achlioptas. Bad global minima exist and sgd can reach them. arXiv preprint arXiv:1906.02613, 2019a.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3296–3305, 2019b.

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, 2019c. 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.

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 Montúfar, Yue Ren, and Leon Zhang. Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. arXiv preprint arXiv:2104.08135, 2021.

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.

NVIDIA Inc. NVIDIA Jetson TX2. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, accessed 2019-09-01.

Douglas A Reynolds. Gaussian mixture models. Encyclopedia of biometrics, 741:659–663, 2009.

Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.

Matthias Scholz, Martin Fraunholz, and Joachim Selbig. Nonlinear principal component analysis: neural network models and applications. In Principal manifolds for data visualization and dimension reduction, pp. 44–67. Springer, 2008.

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.

Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. arXiv preprint arXiv:2009.11094, 2020.

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.

Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. arXiv preprint arXiv:2006.05467, 2020.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376, 2020.

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.

A Algorithm for searching Spline EB tickets

Algorithm 1: The Algorithm for Searching Spline EB Tickets

Initialize the weights $\boldsymbol{W}$ and the FIFO queue Q with length l;
while t (epoch) $< t_{max}$ **do**
  Update $\boldsymbol{W}$ and r using SGD training;
  Calculate the distance between the input space partitions of the current and the previous networks, and then add it to Q.
  $t = t + 1$
  if $\mathbf{Max}(Q) < \epsilon$ **then**
    Return $f(x; \boldsymbol{W})$ (Spline EB ticket);
  end if
end while

B Comparison with pruning at initialization baselines

We further supply another three pruning-at-initialization methods, RigL (Evci et al., 2019), GraSP (Wang et al., 2020), and SynFlow (Tanaka et al., 2020), as our baselines and present the comparison results as shown in Tab. 4. We can see that our method consistently outperforms the pruning-at-initialization baselines in terms of accuracy-efficiency trade-offs, achieving up to 2.68% accuracy improvement at comparable or even (up to 3.2×) lower total training FLOPs.

Table 4: Evaluating our global spline pruning method over SOTA pruning at initialization methods on CIFAR-10/100 datasets. All accuracies are averaged over three runs.

Setting	Methods	Retrain Accuracy (%)			FLOPs (P)		
		p=30%	**p=50%**	**p=70%**	**p=30%**	**p=50%**	**p=70%**
PreResNet-101 CIFAR-10	RigL	93.56	93.26	92.72	7.81	7.81	7.81
	GraSP	93.21	92.83	92.57	7.5	7.5	7.5
	SynFlow	93.25	92.86	91.76	7.5	7.5	7.5
	Spline	94.13	93.92	92.06	13.6	12.1	10.1
	EB Spline	93.67	93.18	92.32	6	4.26	2.74
	Spline Improv.	**+0.57% ~ +0.92%**	**+0.66% ~ +1.09%**	**-0.4% ~ +0.56%**	**0.6× ~ 2.9×**		
VGG-16 CIFAR-10	RigL	92.65	92.72	92.4	15.6	15.6	15.6
	GraSP	92.49	91.47	90.79	15.3	15.3	15.3
	SynFlow	92.74	91.89	92.17	15.3	15.3	15.3
	Spline	93.62	93.46	92.85	26.4	25	19.6
	EB Spline	93.28	93.05	91.96	19.4	15.5	11.1
	Spline Improv.	**+0.88% ~ +1.13%**	**+0.74% ~ 1.99%**	**+0.45% ~ 2.06%**	**0.6× ~ 1.4×**		
PreResNet-101 CIFAR-100	RigL	72.55	71.87	69.92	7.81	7.81	7.81
	GraSP	72.09	71.66	69.6	7.5	7.5	7.5
	SynFlow	72.33	71.88	69.86	7.5	7.5	7.5
	Spline	73.79	72.04	68.24	12.6	10.9	9.36
	EB Spline	72.67	71.99	69.74	5.44	3.84	2.46
	Spline Improv.	**+1.24% ~ +1.7%**	**+0.16% ~ +0.38%**	**-0.18% ~ 0.14%**	**0.6× ~ 3.2×**		
		p=10%	p=30%	p=50%	p=10%	p=30%	p=50%
VGG-16 CIFAR-100	RigL	71.4	70.98	70.75	15.6	15.6	15.6
	GraSP	69.5	69.25	68.43	15.3	15.3	15.3
	SynFlow	72.08	71.48	71.2	15.3	15.3	15.3
	Spline	72.18	71.54	70.07	28	22.9	19.4
	EB Spline	72.07	71.46	70.29	19.9	15.3	11.8
	Spline Improv.	**+0.1% ~ +2.68%**	**+0.06% ~ +2.29%**	**-0.91% ~ +1.86%**	**0.6× ~ 1.3×**		

C Global Spline pruning with FA dimension reduction

We further adopt factor analysis (FA) dimension reduction methods on PreResNet-101 and VGG-16. The results are shown in the Tab. 5. We can see that our spline based pruning method is not sensitive to

Table 5: Global spline pruning with PCA and FA dimension reduction methods.

Setting	Methods	Accuracy (%)		
		p=30%	**p=50%**	**p=70%**
PreResNet-101@CIFAR-10	Spline (FA)	94.27	94.26	91.97
	Spline (PCA)	94.13	93.92	92.06
VGG-16@CIFAR-10	Spline (FA)	93.10	93.25	92.49
	Spline (PCA)	93.62	93.46	92.85
PreResNet-101@CIFAR-100	Spline (FA)	74.37	72.94	68.02
	Spline (PCA)	73.79	72.04	68.24
		p=10%	**p=30%**	**p=50%**
VGG-16@CIFAR-100	Spline (FA)	72.20	71.99	70.62
	Spline (PCA)	72.18	71.54	70.07

the adopted dimension reduction methods as long as it can be used to measure the correlation between two units. We think that studying the theoretical guarantees on what methods and in which regime those methods are in fact preserving that information would be an interesting future research direction to provide further theoretical guarantees.

D How universal is the EB for different optimization methods?

We detect EB tickets using different optimization methods, including SGD, Adam, Adagrad, and RMSprop (Ruder, 2016), and report both the emerging epochs and the retraining accuracies of detected EB tickets on PreResNet-101 and VGG-16 in the Tab. 6. The results show that our spline EB tickets consistently emerge at early training stages and perform on par with their unpruned counterparts, and thus are empirically observed to be universal to different optimization methods.

Table 6: EB tickets detection using different optimization methods.

Setting	Methods	EB Emerge Epoch	Retrain Accuracy (%)		
			Unpruned	**p=30%**	**p=50%**
PreResNet-101@CIFAR-10	EB (SGD)	61	93.66	93.67	93.18
	EB (Adam)	30	89.63	89.63	89.26
	EB (Adagrad)	33	90.64	90.76	90.79
	EB (RMSprop)	80	87.41	86.83	86.37
VGG-16@CIFAR-10	EB (SGD)	24	92.71	93.28	93.05
	EB (Adam)	41	90.11	91.25	90.73
	EB (Adagrad)	27	90.17	91.04	89.55
	EB (RMSprop)	88	87.35	87.86	88.1

E How universal is the EB for adversarial initialization?

we also detect EB tickets with the adversarial initialization (Liu et al., 2019a). The results shown in the Tab. 7 demonstrate that our EB tickets can consistently be found under the adversarial initialization and perform on par with their corresponding unpruned dense networks after being retrained.

F Background of MASO and its ties with our method

The entire CPA/max-affine spline formulation of deep networks (DNs) has been extensively studied before (Balestriero & Baraniuk, 2018). However the prior published works merely focus on studying the various spline properties that one can obtain on a DN from the MASO formulation. Those works did not study the contributions that we propose in this paper, i.e., (1) bridging the connection between spline theory

Table 7: EB tickets detection using adversarial initialization.

Setting	Methods	EB Emerge Epoch	Retrain Accuracy (%)		
			Unpruned	p=30%	p=50%
PreResNet-101@CIFAR-10	EB (Random Init.)	61	93.66	93.67	93.18
	EB (Adv. Init)	81	93.33	93.38	93.36
VGG-16@CIFAR-10	EB (Random Init.)	24	92.71	93.28	93.05
	EB (Adv. Init)	40	92.20	92.63	91.66

and network pruning techniques, (2) discovering that a DN's spline mappings exhibit an early-bird (EB) phenomenon whereby the spline's partition converges at early training stages, and (3) leveraging the aforementioned EB finding to develop a principled pruning strategy that focuses on a tiny fraction of DN nodes whose corresponding spline partition regions actually contribute to the final decision boundary.

Needless to say that, we do not claim that the MASO formulation is one of our contribution. In fact, we refer the readers to those prior works in Sec. 2. Instead, we propose to build-upon the MASO formulation to in-turn study pruning in deep networks, and this (as far as we are aware) has not been done (even succinctly) previously.

G Analysis about the connection between LTH and Spline theory

At the lowest level, LTH and spline theory are connected as follows. A DN partitions its input space according to a power diagram subdivision that is formed by a recursive subdivision process through the layers. Each subdivision is analytically known and involves the layer weights of each unit and the biases. When pruning a unit at layer l, the subdivisions of layers $1, \cdots, l-1$ are unchanged, the power diagram of layer l is altered only by removal of a single hyperplane from its boundary, and all the subdivisions of layers $l+1, \cdots, L$ are altered.

Additionally, in classification tasks, the decision boundary is constrained to be linear within each region of this input space partition. Hence, for the decision to be nonlinear in a part of the space, the input space partition at that location must contain at least two regions. The LTH, in terms of splines, states that it is possible to obtain an input space partition obtained by a power diagram subdivision each containing significantly less regions, that can still be positioned such that the decision boundary solves the task at hand. This is the exact phenomenon we tried to highlight in Fig. 2 and Fig. 3 where we depicted that the same (or similar) decision boundary (in red) could be obtained from a much reduced input space partition.

While we have limited ourselves to mostly empirical evaluations and validations as an important first step, we agree that an in-depth theoretical study would be immensely beneficial for the community. This is something we are hoping to achieve in a next study as this theoretical question is incredibly challenging, for which this work can provide critical insights and inspirations.

H Additional Visualization

We supply the additional visualization of spline trajectories to Fig. 8.

I DN initialization: an alternative to pruning

The initialization dilemma and the importance of overparameterization. In the case of DNs, most initialization techniques focus on maintaining feature maps statistics bounded through depth 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 say kmeans++ remains to be developed for DNs. Hence, overparametrization allows successful training, and a posteriori, one can remove the redundant parameters and obtain a final model with much better performances versus the non-overparametrized and non-pruned counterpart. This is the key motivation of Early Bird tickets. Furthermore, the parallel between DNs and K-means is most relevant as it has been shown

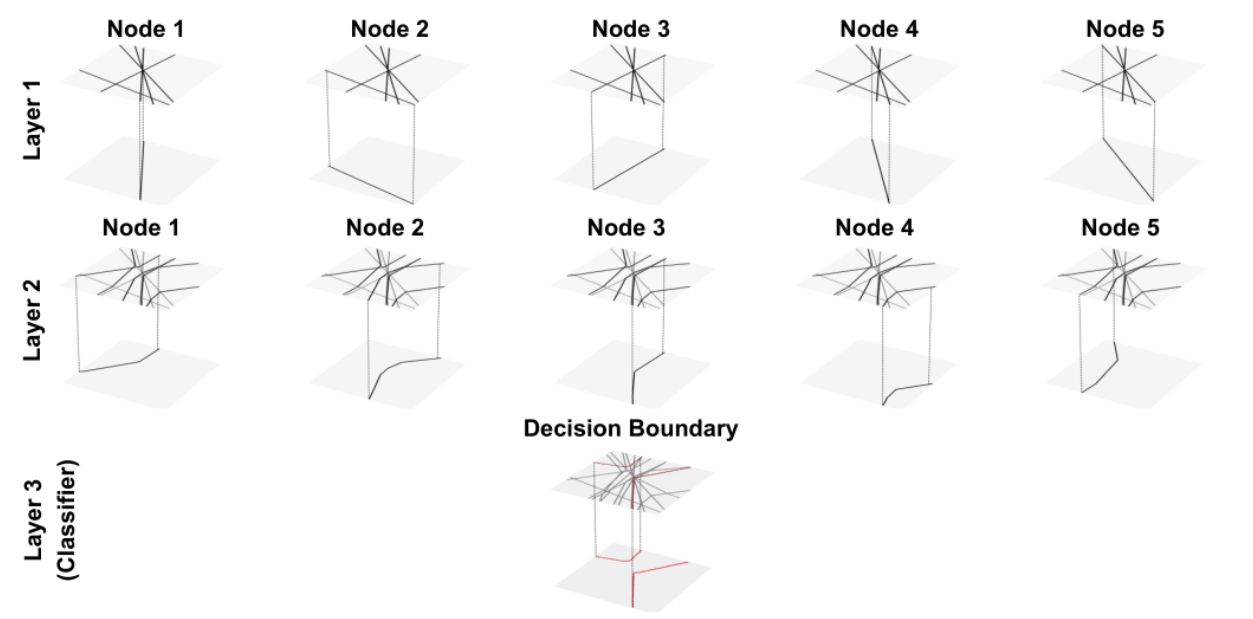

Figure 8: Additional visualization of the partitioning and subdivision layer after layer, where each node introduces a spline in the input space which is depicted under the current partitioning with a highlighted path linked via a dotted line.

in (Balestriero et al., 2019) that the DN decision process relies on an input space partition based on centroids that is very similar to the one of K-means and which thus benefit in the same way to overparametrization. Beyond this geometric aspect, overparametrization has been proven to facilitate optimization (Arpit & Bengio, 2019) and to position the initial parameters close to good local minima (Allen-Zhu et al., 2019; Zou & Gu, 2019; Kawaguchi et al., 2019) reducing the number of updates needed during training.

Remark 1. *Winning tickets are the result of employing overly parametrized DNs which are simpler to optimize and produce better performances, as current optimization techniques can not escape from poor local minima and advanced DN initialization (near good local minima) is unknown.*

We further support the above remark in the following paragraphs where we demonstrate how the absence of good initialization coupled with non-optimal optimization problems impacts performances unless overparametrization is used, in which case winning tickets naturally emerge.

DN initialization alternative: layerwise pretraining. We saw in the previous section that the concept of winning tickets emerges from the need to overparametrize DNs which in turn emerges naturally from architecture search and cross-validation as overparametrizing greatly facilitates training and improves final results. We now show that if a better initialization of DNs existed, one would have the ability to train a minimal DN directly and thus would not resort to the entire pruning pipeline.

We convey the above point with a carefully designed experiment. We consider three cases. First, the case of employ a minimal DN with random weights initialized from random Kaiming initialization (He et al., 2015). Second, we consider the same minimal DN architecture but with weights initialized based on unsupervised layerwise pretraining which we consider as a data-aware initialization (no label information is used) (Belilovsky et al., 2019). In both cases, training is done on the classification task in the same manner. Third, we consider an overparametrize DN trained with the lottery ticket (LT) method (training, pruning, and re-training). The final models of the three cases have the same architecture (but different weights based on their own training method). We report their classification results in Table 8, from which we can see that especially for very small final DNs (high pruning ratios) LT models outperform a randomly initialized DN, but in turn a well initialized DN is able to outperform LT training. From this, we see that *the ability of*

Table 8: Accuracies of layerwise (LW) pretraining, structured pruning with random and lottery ticket initialization.

Setting	Pruning Ratio	Random Init.	Lottery Init.	LW Pretrain
VGG-16 on CIFAR-10	30%	93.33±0.01	**93.57±0.01**	93.08±0.00
	50%	93.07±0.03	**93.55±0.03**	93.08±0.01
	70%	92.68±0.02	**93.44±0.01**	92.81±0.02
	90%	90.48±0.06	90.41±0.23	**90.88±0.02**
VGG-16 on CIFAR-100	10%	71.49±0.03	**71.70±0.09**	71.14±0.02
	30%	71.34±0.10	71.24±1.18	**71.35±0.01**
	50%	67.74±1.05	69.73±1.15	**70.19±0.01**
	70%	60.44±4.98	66.61±0.95	**67.40±0.83**

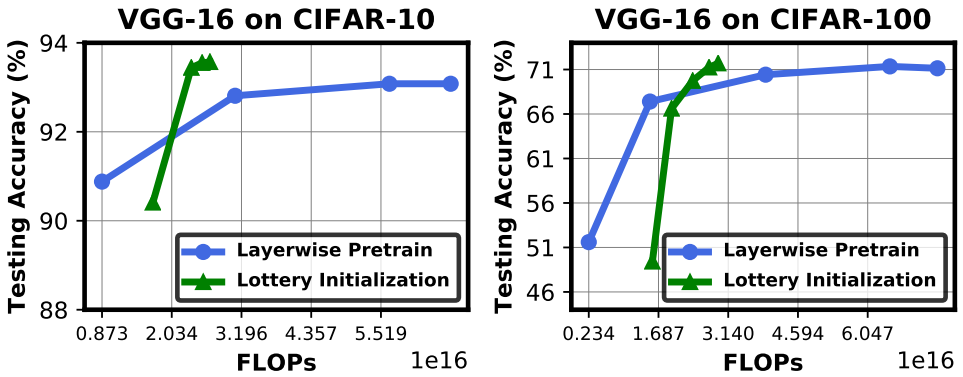

Figure 9: Accuracy vs. efficiency trade-offs of lottery initialization and layerwise pretraining.

pruning methods and, in particular, LT to produce better-performing minimal DNs than directly training the same minimal DN lies in the 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 LT training, As shown in Fig. 9, we further compare the required FLOPs when networks are initialized using lottery initialization and layerwise pretraining, respectively. We observer that (1) when the pruning ratio is low (i.e., $<$ 50%), networks with lottery initialization require a smaller number of computational FLOPs to provide a good initialization for the pruned network, while leading to a comparable or even higher retraining accuracy; and (2) when the pruning ratio is higher, layerwise pretraining requires a much smaller number of computational FLOPs as compared to training highly overparametrized dense networks. Such a phenomenon opens a door for investigating the following two questions, which we leave as our 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)?

As the amount of different architectures grows rapidly and the specificity of those architectures can vary drastically, simple layerwise pretraining falls short of providing an advanced initialization solution. For example, it is not clear how layerwise pretraining can be used with a DenseNet Huang et al. (2017) where some parameters connect layers that are far apart in the architecture. Hence, while we believe in searching for improved initialization strategies, we now focus on studying LT training and DN pruning as they provide a universal solution.

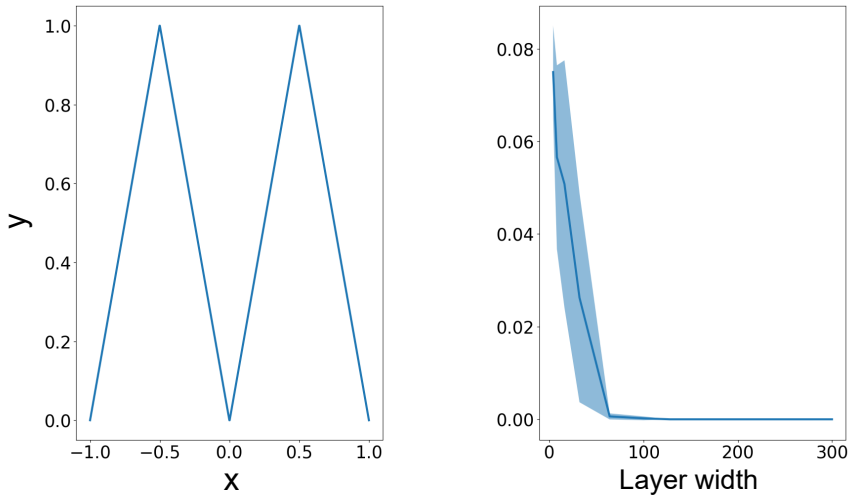

Figure 11: **Left**: Depiction of a simple (toy) univariate regression task with target function being a sawtooth with two peaks. **Right**: The ℓ_2 training error (y-axis) as a function of the width of the DN layer (2 layers in total). In theory, only 4 units are requires to perfectly solve the task at hand with a ReLU layer, however we see that optimization in narrow DNs is difficult and gradient based learning fails to find the correct layer parameters. As the width is increased as the difficulty of the optimization problem reduces and SGD manages to find a good set of parameters solving the regression task.

J Analysis about why there are redundant partition boundaries

Network pruning literature follows the chaining of *(i) over-parametrization, (ii) training, and (iii) pruning* to obtain a small but critical subnetwork, i.e., "winning ticket" that achieve high accuracies. This offers a powerful alternative to training a small network from scratch as good initialization for such sparse network is not known (Frankle & Carbin, 2019a; Blalock et al., 2020) making the optimization challenging. This strategy finds benefits not only with DN but also with traditional methods such as K-means, in which case pruning removes centroids (Kanungo et al., 2002; Hamerly & Elkan, 2002; Celebi et al., 2013). We propose to briefly employ K-means to demonstrate the superiority of such pruning strategies. As shown in Fig. 10, we consider K-means++ (Arthur & Vassilvitskii, 2006) as a ground-truth method to represent good initialization (denoted as kmeans++), where we know a priori the number of clusters (64) for artificial data generated from a Gaussian Mixture Model (GMM) (Reynolds, 2009) with spherical and identical covariances. Against that baseline, we perform the three-step pruning strategy over multiple runs and with varying numbers of initial clusters to generate winning initialization (denoted as tickets). In all case the clusters will be pruned to 64 (true number of Gaussians) from the initial number that varies from 64 to 128 ($\boldsymbol{x}$-axis of Fig. 10). We observe that the winning initialization perform near-optimal results (i.e., closer to kmeans++) when starting from overparameterization (i.e., initial clusters $\geq$ 100), otherwise suffer from large accuracy drops. We highlight that employing overparameterization facilitate finding winning tickets. From the above experiments it is clear that in the absence of "optimal" initialization of small DNs, pruning is the current preferred solution to obtain performing minimal architectures.

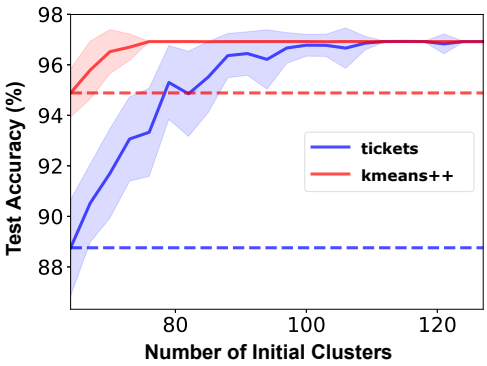

Figure 10: K-means experiments on a toy mixture of 64 Gaussian.

K Additional results on initialization and pruning

Next we extend the overparametrization-pruning vs. initialization insights to univariate DNs on a carefully designed dataset. Considering a simple unidimensional sawtooth as displayed in Fig. 11 with P peaks (here $P = 2$). In the special case of a single hidden layer with a ReLU activation function, one must have at least $2P$ units to perfectly fit this function with the weight configuration being $[\boldsymbol{W}_1]_{1,k} = 1, [\boldsymbol{b}_1]_k = -k$ with $k = 1, \ldots, D_1$ and $[\boldsymbol{W}_2]_{1,1} = 1, [\boldsymbol{W}_2]_{1,k}(-2)^{(k-1)\%2}, k = 2, \ldots, D_1$. Note that these 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, the probability that a random weight initialization arranges the initial splines in a way to allow effective gradient based training 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 empirically illustrated in Fig. 11 (right) where one can see that even repeating multiple initializations of a DN without overparametrization does not allow to solve the task, while overparametrizing, training, and then pruning that together preserve only the correct number of units allow for better approximation.

L Relation between activation patterns and spline partition of input space

Subdivision Lines. Every layer in deep networks (DNs) with piecewise nonlinearities (e.g. ReLU activation) can be viewed as subdivision lines for partitioning the given input space (Balestriero et al., 2019). For example, suppose the DNs' input space is shaped as a square grid with $N \times N$ data points, we extract the activations from one intermediate layer of k hidden nodes, then each node corresponds one subdivision line to distinguish non-zero activations in the input grid, thus we have k subdivision lines in this layer, same as the number of hidden nodes. From such geometry perspective, there are two good characteristics to leverage: (1) Subdivision lines in the first layer are linear affine functions taking layer parameters as slopes and biases, following subdivision lines are piecewise linear, whose turning points are exactly located at the subdivision lines in previous layers; (2) The derived subdivision lines at the final classification layer are exactly the network decision boundary. Such connection provides us a new perspective to analyze *how the decision boundary are gradually formulated* and *what network compression methods (e.g., pruning, quantization) really mean from such geometry perspective.*

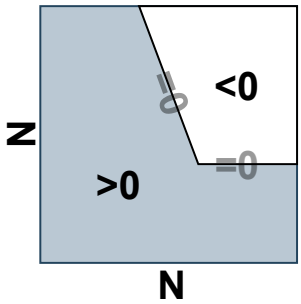

Figure 12: Visualize one subdivision line on grid.

M Early-bird visualization on test and random samples

To investigate what happens to the partitioned regions that do not contain any training data, we redraw Fig. 5 (early-bird visualization) on test samples, and show the visualization of early-bird phenomenon at Fig. 13.

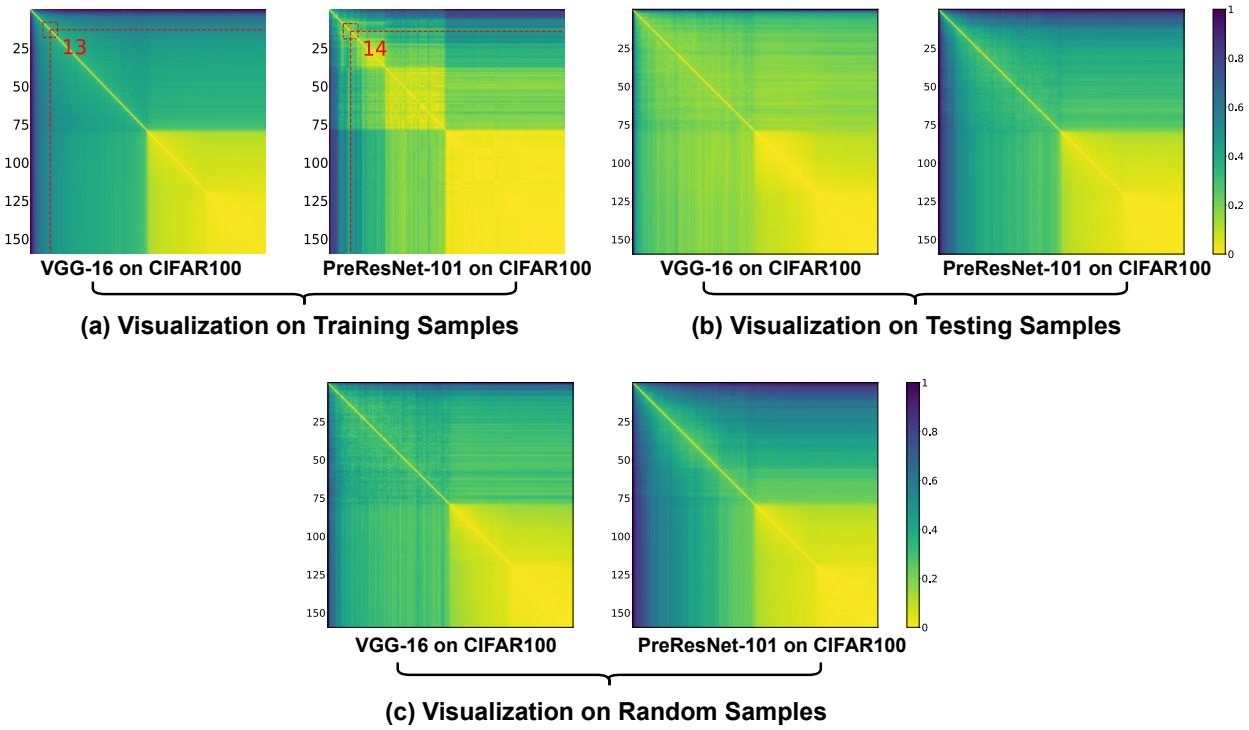

Figure 13: visualization of the early-bird (EB) phenomenon when using training samples, testing samples, and random samples, respectively.

N Spline trajectory and EB tickets with Leaky ReLU activation

We redraw Fig. 4 (spline trajectory) and Fig. 5 (early-bird visualization) when using Leaky ReLU as activation functions, and show the corresponding spline trajectory at Fig. 14 and the visualization of early-bird phenomenon at Fig. 15. From these figures, we can consistently observe that the splines will adapt during the early phase while converging at later training stages, demonstrating that the early-bird phenomenon still holds under the Leaky ReLU activation functions. In addition, we supply the pruning experiment comparisons using networks with Leaky-ReLU activation functions to Table 9, from which we see that the spline pruning consistently outperforms all baselines in terms of both accuracy and efficiency, leading to -0.44% ~ +1.38% accuracy improvement and up to 4× training flops savings.

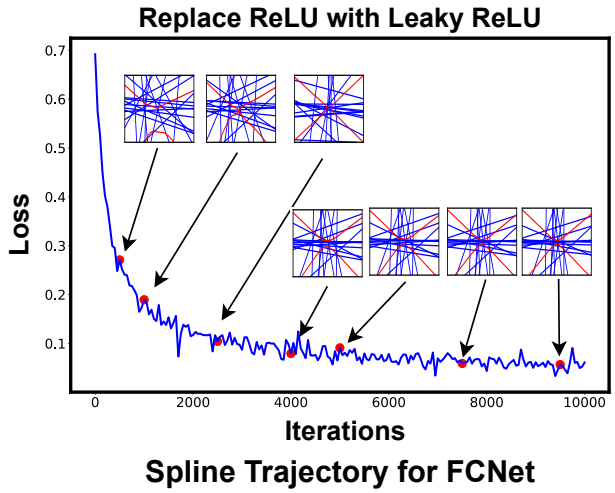

Figure 14: Visualization of spline trajectories using FCNet with Leaky ReLU activation functions.

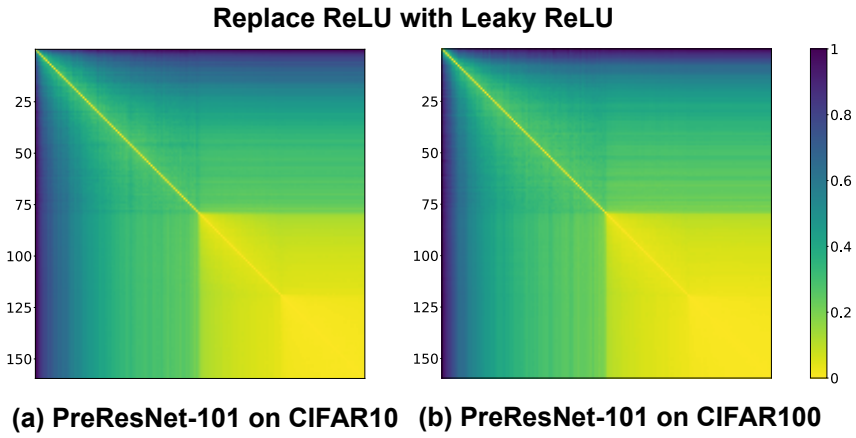

Figure 15: Visualization of the early-bird (EB) phenomenon when training PreResNet-101 with Leaky ReLU activation functions.

Table 9: Evaluating our spline pruning method over baselines when using Leaky ReLU activation functions.

Setting	Methods	Retrain Accuracy (%)			FLOPs (P)		
		p=30%	**p=50%**	**p=70%**	**p=30%**	**p=50%**	**p=70%**
PreResNet-101 on CIFAR-10	NS	93.54%	93.18%	90.89%	13.9P	12.7P	11.0P
	ThiNet	92.18%	91.57%	90.25%	13.2P	10.6P	8.65P
	Spline	93.56%	92.74%	91.11%	13.6P	12.1P	10.1P
	EB Spline	92.76%	92.13%	90.66%	6.00P	4.26P	2.74P
	Spline Improv.	**-0.44% ~ +1.38%**			**1.1x ~ 4.0x**		

O Comparison under 90% pruning ratios

As shown in the Table 10, the proposed spline pruning outperforms the baselines across two models (PreResNet-101 and VGG-16) and two datasets (CIFAR-10 and CIFAR-100), leading to -0.35% ~ +63.87% accuracy improvements. Note that the network slimming (NS) method suffers from bottleneck layers due to the "over-pruning" channels on specific layers.

Table 10: Evaluating our spline pruning method over baselines under 90% pruning ratios.

Datasets	Model	Methods	Accuracy (p=90%)
CIFAR-10	PreResNet-101	NS	77.63%
		ThiNet	87.80%
		Spline	87.96%
		EB Spline	88.15%
		Spline Improv.	**+0.35% ~ +10.52%**
	VGG-16	NS	89.13%
		ThiNet	86.90%
		Spline	89.11%
		EB Spline	89.24%
		Spline Improv.	**+0.11% ~ +2.34%**
CIFAR-100	PreResNet-101	NS	28.90%
		ThiNet	60.66%
		Spline	58.97%
		EB Spline	60.31%
		Spline Improv.	**-0.35% ~ +31.41%**
	VGG-16	NS	1%
		ThiNet	56.31%
		Spline	59.37%
		EB Spline	64.87%
		Spline Improv.	**+8.56% ~ 63.87%**

P Comparison with iterative pruning methods

We further supply the comparison with network slimming with the mentioned iterative pruning method (NS-IP). As shown in the Table 11, our spline pruning again outperforms those methods in terms of both accuracy and efficiency, leading to -0.47% ~ +1.11% accuracy improvement and up to 7.1× training flops savings. We will add this set of experiments and discussion in our revision.

Q Quantitative distances of different layers along training trajectory

As shown in the Table 12, the quantitative distance of early layers converges faster than later layers, indicating that the base partition regions divided by first few layers will not change too much since the very early epochs.

Table 11: Evaluating our spline pruning method over iterative pruning baseline.

Setting	Methods	Retrain Accuracy (%)			FLOPs (P)		
		p=30%	**p=50%**	**p=70%**	**p=30%**	**p=50%**	**p=70%**
PreResNet-101 on CIFAR-10	NS-IP	93.74%	92.72%	92.53%	23.6P	21.6P	18.7P
	Spline	94.13%	93.92%	92.06%	13.6P	12.1P	10.1P
	EB Spline	93.67%	93.18%	92.32%	6.00P	4.26P	2.74P
	Spline Improv.	**+0.39%**	**+1.20%**	**-0.47%**	**1.7x ~ 6.8x**		
PreResNet-101 on CIFAR-100	NS-IP	72.68%	71.82%	69.86%	23.3P	21.3P	17.5P
	Spline	73.79%	72.04%	68.24%	12.6P	10.9P	9.36P
	EB Spline	72.67%	71.99%	69.74%	5.44P	3.84P	2.46P
	Spline Improv.	**+1.11%**	**+0.22%**	**-0.12%**	**1.8x ~ 7.1x**		

Table 12: Record of the quantitative distance between input space partitions of different layers, i.e., early, middle, or later layers, along the training trajectory.

Datasets	Models	Layers	Quantitative Distances				
			10th epoch	**40th epoch**	**80th epoch**	**120th epoch**	**160th epoch**
CIFAR-100	VGG-16	Early (1st)	0.247	0.197	0.061	0.021	0.003
		Medium (8th)	0.538	0.424	0.183	0.024	0.003
		Late (16th)	0.579	0.512	0.236	0.027	0.004
	PreResNet-101	Early (1st)	0.621	0.562	0.296	0.023	0.002
		Medium (16th)	0.704	0.612	0.273	0.025	0.003
		Late (32th)	0.781	0.728	0.376	0.038	0.005