Finding Lottery Tickets in Vision Models via Data-driven Spectral Foresight Pruning Supplementary Material

A. Implementation Details

In Table 1 you can find the training details used in this work. We evaluate each algorithm on `trivial` (36.00%, 59.04%, 73.80%), `mild` (83.22%, 89.30%, 93.12%) and `extreme` (95.60%, 97.17%, 98.20%) sparsity ratios as [21]. In each experiment, we use 100 rounds for iterative PaI methods adopting an exponential schedule as [8, 19]. We train and test on the respective official splits of each dataset, repeating each experiment three times.

Classification - Random initialization. For the classification experiments starting from Kaiming Normal initialization [10], we follow [19, 21]. The augmentations used when training on CIFAR-10 and CIFAR-100 [12] are Random Crop to 32×32 with padding 4 followed by Random Horizontal Flipping with 0.5 probability. For the experiments on Tiny-ImageNet [5], we augment the training images with Random Resized Crop to 64×64 with scaling going from 0.1 to 1.0 using 0.8 x-ratio and 1.25 y-ratio. Then, we apply Random Horizontal Flipping with 0.5 probability. On ImageNet [5], we apply Random Resized Crop to 224×224 with scaling going from 0.2 to 1.0 using 3/4 x-ratio and 4/3 y-ratio. Then, we apply Random Grayscaling with 0.2 probability, Color Jitter with brightness, contrast, saturation and hue all set to 0.4. Finally, we apply Random Horizontal Flipping with 0.5 probability.

Classification - Pre-trained models. Regarding the classification experiments when starting from ImageNet [5], MoCov2 on ImageNet [4] and CLIP [16] pre-trained models, we align with [3]. Specifically, we use the same augmentations detailed in the previous paragraph but we adjust the cropping and rescaling transformations to ensure that the resultant image size is set at 224×224 pixels, aligning with the dimensions of the images used in obtaining the pre-trained models.

Segmentation. For the semantic segmentation experiments we again align with [3]. We employ the following augmentations during training: Random Scale with a range between 0.5 and 2.0, Random Crop to 513×513, followed by Random Horizontal Flipping with 0.5 probability.

Pre-trained models & Architectures. Regarding the pre-trained models used in our experiments, we employed the official ImageNet pre-trained model from the PyTorch torchVision package [14]. The MoCov2 ImageNet model we used is the official one from Facebook research[1]. The CLIP pre-trained model is the official one from OpenAI[2]. Finally, we base our experiments on DINO [1] from its officially released pre-trained model[3].

Our code is based on the framework for Pruning-at-Initialization provided by [19]. Moreover, we used their implementations for the architectures used in our classification experiments. For the segmentation experiments, we align with [3] and use the same implementation of DeepLabV3+[4].

Choice of the pruning set. To perform the foresight pruning procedure using data-driven methods we employ a pruning data split composed of ten examples per class, in line with the work of [13, 21]. For the data-free strategies, we use an equal amount of mini-batches.

B. Additional Discussions

In this section we provide the derivations and intuitions about the mathematics used in the main submission, to make it clear how the path-wise perspective can be studied via forward and backward passes on any architecture. Note that in all of our derivations and formulas, we skip bias terms as we embed them in the weight matrix by adding an additional input set to 1 to each neuron.

Frobenius norm of the Path Activation matrix. In Eq. (7) of the main submission we applied the definition of the Frobenius norm on the Path Activation matrix

$$\|\boldsymbol{J}_{\boldsymbol{v}}^{\boldsymbol{f}}(\boldsymbol{X})\|_F^2 = \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{p=1}^{P}\left(\sum_{s=1}^{d}\mathbb{I}[p\in\mathcal{P}_{s\to k}]a_p(\boldsymbol{x}_n,\boldsymbol{\theta})\boldsymbol{x}_{n_s}\right)^2$$

We observe that by fixing a path p, the inner sum from $s=1$

[1] https://github.com/facebookresearch/moco
[2] https://github.com/openai/CLIP
[3] https://github.com/facebookresearch/dino
[4] https://github.com/VainF/DeepLabV3Plus-Pytorch

Dataset	Architecture	Optimizer	LR	LR Drop	Momentum	Weight Decay	Epochs	Batch Size
Classification - Kaiming Normal initialization [10]								
CIFAR-10 [12]	ResNet-20 [11]	SGD [17]	0.1	x10 at epochs 80, 120	0.9	1e-4	160	128
CIFAR-100 [12]	VGG-16 [18]	SGD (Nesterov) [17]	0.1	x10 at epochs 60, 120	0.9	5e-4	160	128
Tiny-ImageNet [5]	ResNet-18 [11]	SGD [17]	0.2	x10 at epochs 100, 150	0.9	1e-4	200	256
ImageNet [5]	ResNet-50 [11]	SGD [17]	0.1	x10 at epochs 30, 60, 80	0.9	1e-4	90	448
Classification - ImageNet [5], MoCov2 on ImageNet [4], CLIP [16] pre-trained models								
CIFAR-10 [12]	ResNet-50 [11]	SGD [17]	0.1	x10 at epochs 91, 136	0.9	1e-4	182	256
CIFAR-100 [12]	ResNet-50 [11]	SGD [17]	0.1	x10 at epochs 91, 136	0.9	1e-4	182	256
Tiny-ImageNet [5]	ResNet-50 [11]	SGD [17]	0.1	x10 at epochs 91, 136	0.9	1e-4	182	256
Segmentation - ImageNet [5], MoCov2 on ImageNet [4], DINO on ImageNet [1] pre-trained models								
Pascal VOC2012 [6]	DeepLabV3+ (ResNet-50) [2]	SGD [17]	0.001	x10 at epochs 50, 60	0.9	1e-4	80	4

Table 1. Training setups used in this work.

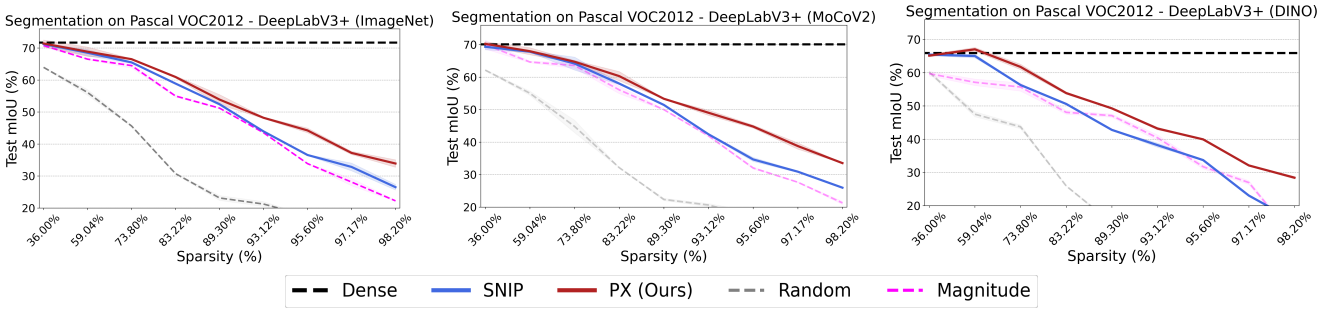

Figure 1. Average mean Intersection over Union (mIoU) at different sparsity levels on Pascal VOC2012 using DeepLabV3+ with pre-trained ResNet-50 as the backbone. Each experiment is repeated three times. Standard deviations are in shaded colors.

Pruning Method	Computational Complexity	Epochs to Reach 98.20% Sparsity	Accuracy at 98.20% Sparsity
IMP [7]	$\mathcal{O}(1)$	960	77.38
Single-shot PaI methods			
Random	$\mathcal{O}(1)$	0	72.31
Magnitude	$\mathcal{O}(1)$	0	76.12
SNIP [13]	$B \cdot ([FP] + [BP])$	0	75.39
GraSP [20]	$B \cdot (2[FP] + 2[BP])$	0	76.30
Iterative PaI methods			
SynFlow [19]	$T \cdot ([FP] + [BP])$	0	75.19
NTK-SAP [21]	$T \cdot B \cdot (3[FP] + [BP])$	0	74.55
PX (Ours)	$T \cdot B \cdot (3[FP] + [BP])$	0	77.08

Table 2. Comparison of the computational complexity of each pruning procedure. Last column is the accuracy at 98.20% sparsity when starting from ResNet-20 on CIFAR-10.

to d will have only one non-zero term (given by the indicator function). Specifically, the one for which s is the starter input node: the other input nodes cannot take part in path p as the first connection will define a different path. This means that the equality reported in Eq. (7) of the main submission holds, provided that we take the correct input node $s|s \in p$.

From layer-wise to path-wise. Here we provide the key idea on how to pass from the layer-wise perspective to the path-wise perspective when considering activations $\boldsymbol{a} \in \mathbb{R}^m$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^m$. First of all, the relationship between $\boldsymbol{a}$ and a_p is given by the definition of the latter. According to what we reported in Section 3.3 of the main submission, $a_p(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{\{i|\theta_i \in p\}} \mathbb{I}[z_i > 0]$ where z_i is the activation of the neuron connected to the previous layer through parameter θ_i. Thus, a_p is a binary value representing the activation status of path p. This is equivalent to assigning a binary mask to each paramenter $\boldsymbol{\theta}_j$ representing the activation status of the neuron that it connects to (*i.e.* $\boldsymbol{a}_j(\boldsymbol{x}, \boldsymbol{\theta}) = \mathbb{I}[z_j > 0]$). This resulting binary-valued mask is $\boldsymbol{a}$. The expression of the k-th output of a neural network, can be written as [9]

$$
\begin{aligned}
\boldsymbol{f}^k(\boldsymbol{x}, \boldsymbol{\theta}) &= \sum_{s=1}^{d} \sum_{p \in \mathcal{P}_{s \to k}} \boldsymbol{v}_p(\boldsymbol{\theta}) a_p(\boldsymbol{x}, \boldsymbol{\theta}) \boldsymbol{x}_s \\
&= \sum_{s=1}^{d} \sum_{p \in \mathcal{P}_{s \to k}} \prod_{j | j \in p} \boldsymbol{\theta}_j \boldsymbol{a}_j(\boldsymbol{x}, \boldsymbol{\theta}) \boldsymbol{x}_s \\
&= \left[\prod_{l=1}^{L} \boldsymbol{\theta}^{[l]} \boldsymbol{a}^{[l]}(\boldsymbol{x}, \boldsymbol{\theta}) \boldsymbol{x} \right]_k
\end{aligned}
$$

where L is the total number of layers and l identifies the parameters and the activations of the l-th layer. We can recover the last equality observing that the second row of the equation is the definition of M (where M is the length of path p) consecutive matrix multiplications between the inputs and parameters of each layer, followed by an element-wise multiplication with the activations.

Computational cost analysis. This analysis, detailed in Table 2, focuses on understanding the computational complexity of our method in contrast to that of our competitors. Notably, we include IMP in this assessment to offer a broader context to our study. In the second column of the table, we present the computational cost of invoking each pruning

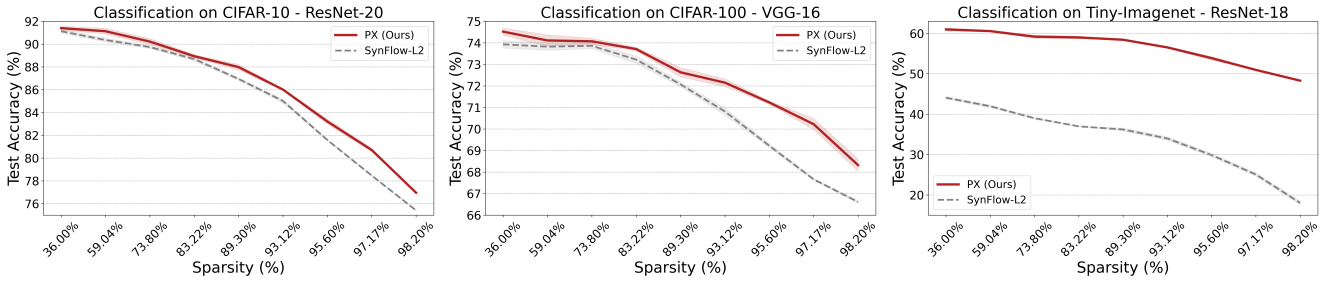

Figure 2. Average classification accuracy at different sparsity levels on CIFAR-10 using ResNet-20, CIFAR-100 using VGG-16 and Tiny-ImageNet using ResNet-18, respectively. Each experiment is repeated three times. We report in shaded colors the standard deviation.

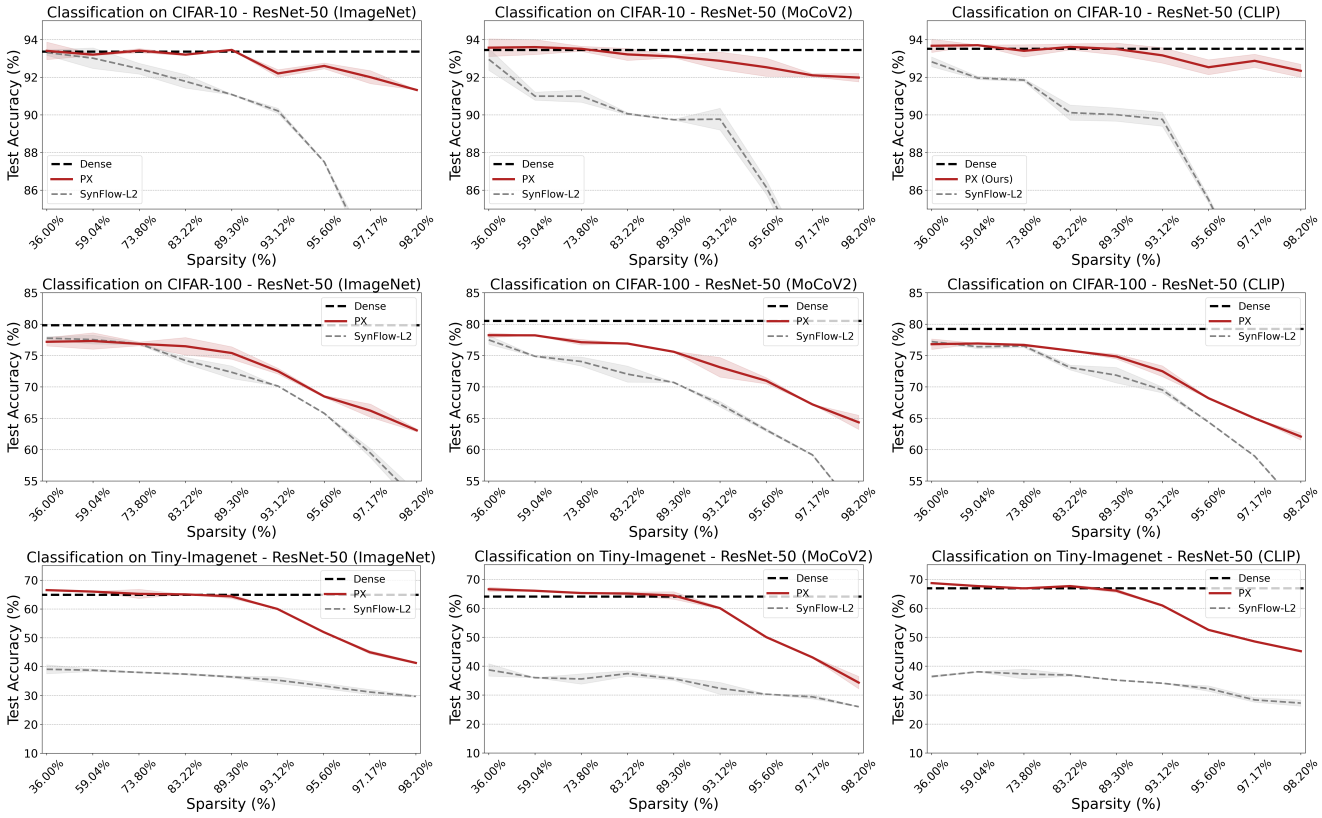

Figure 3. Average classification accuracy at different sparsity levels on CIFAR-10, CIFAR-100 and Tiny-ImageNet using pre-trained ResNet-50 as architecture. The first column reports the results of starting from the supervised ImageNet pre-training. The second column reports the performance when starting from the MoCov2 pre-training on ImageNet. Finally, in the third column we report the results when starting from CLIP. Each experiment is repeated three times. We report in shaded colors the standard deviation.

procedure, measured in numbers of macro-operations performed to obtain the final pruning scores. Where we report $\mathcal{O}(1)$ complexity, it means that the scores can be obtained immediately by simply looking at some intrinsic property of the network, such as the magnitude of the weights, which does not require any additional processing. Here, T represents the required number of pruning iterations, and B indicates the number of mini-batches processed by each algorithm during this procedure. Additionally, we denote the costs of a single forward pass with $[FP]$ and of a single backward pass with $[BP]$. Columns three and four illustrate the training epochs necessary for the pruning algorithm to attain approximately equivalent accuracy at 98.20% sparsity when starting from ResNet-20 on CIFAR-10. This metric trivially stands at zero for PaI methods, given that the procedure is executed prior to training. However, IMP demands a minimum of 6 iterative rounds of pruning and subsequent re-training (each full training cycle spans 160 epochs) to

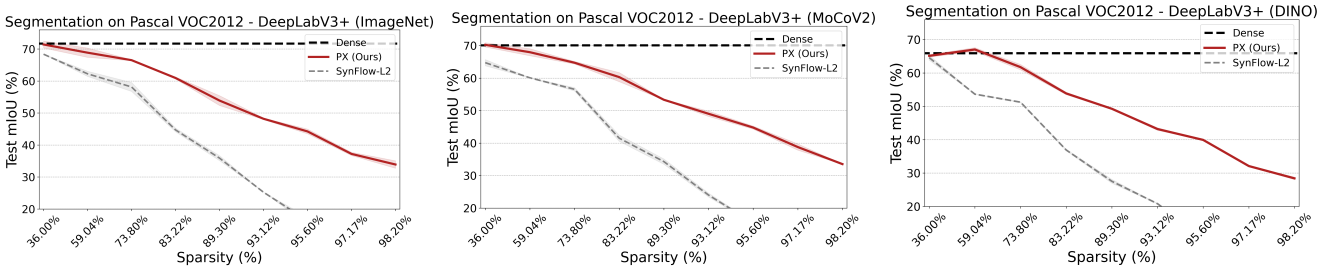

Figure 4. Average mean Intersection over Union (mIoU) at different sparsity levels on Pascal VOC2012 using DeepLabV3+ with pre-trained ResNet-50 as the backbone. Each experiment is repeated three times. Standard deviations are in shaded colors.

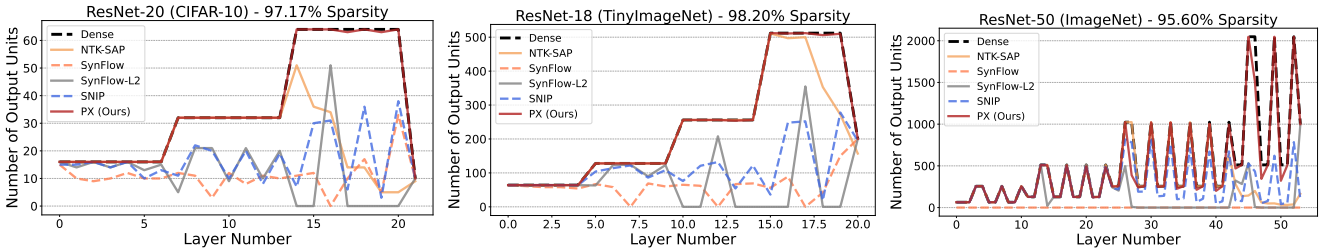

Figure 5. Active output units at different sparsities in ResNet-20, ResNet-18 and ResNet-50. For SNIP and PX data mini-batches are sampled from CIFAR-10, Tiny-ImageNet and ImageNet, respectively.

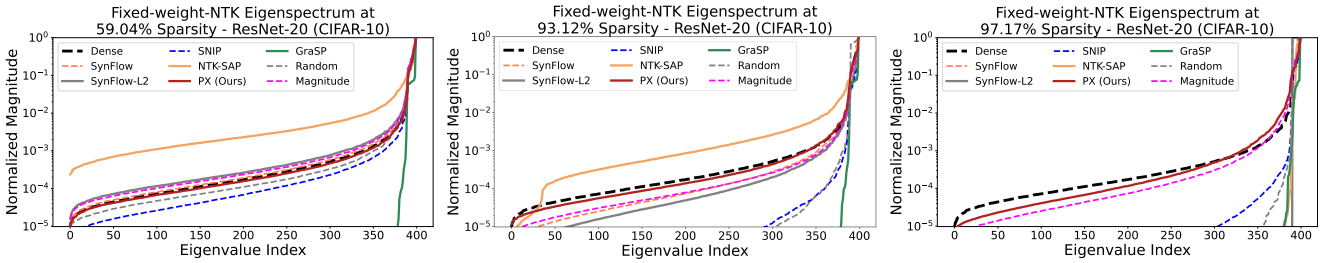

Figure 6. Fixed-Weight-NTK spectrum of ResNet-20 on the CIFAR-10 dataset at different sparsity ratios.

surpass the accuracy reached by our method. Within this analysis, it becomes apparent that PX's computational complexity mirrors that of NTK-SAP [21], aligning generally with iterative PaI methods, which are explicitly constructed to iterate across T rounds preceding training.

Procedure clarifications and pseudocode. In Algorithm 1, we provide the pseudocode to further clarify the role of T and B in the implementation of PX. The functions $\boldsymbol{g}, \boldsymbol{h}$ are copies of $\boldsymbol{f}$ defined for clarity, but the memory usage does not double as PX only stores $\boldsymbol{\theta}$ and the derivatives w.r.t. $\boldsymbol{\theta}^2$ are computed in a single pass (lines 13-15). PX is an iterative PaI method but differs from the standard framework in lines 3, 9-15. We remark that iteratively refining the pruning mask $\boldsymbol{M}$ in T rounds while yielding positive saliency scores guarantees to avoid layer collapse [19].

C. Additional Experiments

Segmentation experiments. In Fig. 1 we report the full results of the semantic segmentation experiments on the Pascal VOC2012 [6] dataset. In each experiment the architecture used is DeepLabV3+ [2] on a ResNet-50 [11] backbone, starting from ImageNet [5], MoCov2 on ImageNet [4] and DINO [1] pre-trained models.

The general trend reported in the main paper is confirmed also in this setting, where our method is able to retain the accuracy of the dense baseline at trivial sparsities.

SynFlow-L2 ablation study. As mentioned in the main paper, our method can be interpreted as an extension of SynFlow-L2 [9]. The core difference is that PX reweights the network's outputs on the basis of the information provided by the data: that information indicates how much each weight contributes to the upper bound on the trace of the NTK reported in Eq. (6) of the main submission. Thus, by comparing PX with SynFlow-L2 we conduct an ablation study to provide further evidence regarding the soundness of the hypotheses underpinning our algorithm, proving the importance of the data-dependent component. In Fig. 2, 3 and 4 we observe that our method is always able to improve over SynFlow-L2. Furthermore, the latter exhibits a

Data: Network $\boldsymbol{f}$ parametrized by $\boldsymbol{\theta}$, pruning dataset $\mathcal{D}$ made of B mini-batches, number of pruning rounds T, final sparsity level k
Result: Parameter mask $\boldsymbol{M}$ used to sparsify $\boldsymbol{f}$ before training

```
# .detach(): the operation is detached from the computational graph.
# z is the output of a network, a is the vector of activations.
g, h = f # create two copies of f
M = 1 # init. parameter mask to all 1s
for t in 1, 2, ..., T do
    # perform the t-th pruning round
    p = k^(t/T) # compute the % of weights to remove at round t
    s = 0 # init. saliency scores to all 0s
    for i in 1, 2, ..., B do
        x = D_i # mini-batch of data at index i
        _, a = f(x, θ ⊙ M, _).detach() # record activations a
        z_g, _ = g(x^2, 1 ⊙ M, a).detach() # force activations to a
        z_h, _ = h(1, θ^2 ⊙ M, 1) # forward an input of all 1s
        R = (z_g ⊙ z_h).sum() # compute score function
        s += R.backward() ⊙ θ^2 # update param.-wise saliency scores
    # update M to keep only top-p parameters
    s̃ = sort_descending(s) # sorted saliency scores in descending order
    p = length(θ) · p # compute top-p threshold index
    for j in 1, 2, ..., length(θ) do
        if s_j − s̃_p < 0 then
            M_j = 0 # set the mask of the j-th param. to zero
return M
```

Algorithm 1: Pruning via Path eXclusion (PX)

drastic decrease in performance when the cardinality of the network's output increases. As already noticed in [15], this is attributed to the combined effect of reducing the layer width while keeping a high path count in the architecture.

Layer widths. In Fig. 5, we present additional plots on the layer width to confirm the trend reported in the main submission regarding the number of output units preserved by PX at each layer. We observe again that PX is able to preserve the output width despite the very high sparsity ratios under exam and the different model sizes.

Spectral analyses. Fig. 6 provides further evidence regarding the preservation of the full eigenspectrum thanks to the approximation of our upper bound detailed in Eq. (6) of the main submission.

Data amount analysis. We indicate with $|D|$ the number of examples per class. The relationship between $|D|$ and the number of batches B is given by $B = \lceil |D| \times |C| / b \rceil$, where $|C|$ is the number of classes, and b is the batch size. In Table 3, we extend Table 2 by showing the effect of changing B and $|D|$ on the data dependent methods. PX is less sensitive than the competitors, and increasing $|D|$ does not yield a significant performance gain in line with the findings of [13].

ResNet-20 (CIFAR-10) - 98.20% Sparsity - ($T = 100$)					
	$B=1$ $\lvert D\rvert=1$	$B=1$ $\lvert D\rvert=5$	$B=1$ $\lvert D\rvert=10$	$B=2$ $\lvert D\rvert=50$	$B=4$ $\lvert D\rvert=100$
Data-driven Single-shot PaI methods					
SNIP [13]	73.01	74.11	75.39	75.17	75.49
GraSP [20]	74.74	76.11	76.30	76.41	76.36
Data-driven Iterative PaI methods					
PX (Ours)	76.96	77.06	77.08	77.16	77.13

Table 3. Ablation on the amount of data used to estimate the saliency scores for the data-driven pruning methods. We report the classification accuracy at 98.20% sparsity when starting from ResNet-20 on CIFAR-10, while varying the amount of examples per class $|D|$.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1, 2, 4

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2, 4

[3] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *CVPR*, 2021. 1

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 4

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2, 4

[6] Mark Everingham, Ali SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 2, 4

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019. 2

[8] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *ICLR*, 2021. 1

[9] Thomas Gebhart, Udit Saxena, and Paul Schrater. A unified paths perspective for pruning at initialization. *arXiv preprint arXiv:2101.10552*, 2021. 2, 4

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 1, 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009. 1, 2

[13] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 1, 2, 5

[14] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 1

[15] Shreyas Malakarjun Patil and Constantine Dovrolis. Phew: Constructing sparse networks that learn fast and generalize well without training data. In *ICML*, 2021. 5

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2

[17] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 2

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[19] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020. 1, 2, 4

[20] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 2, 5

[21] Yite Wang, Dawei Li, and Ruoyu Sun. Ntk-sap: Improving neural network pruning by aligning training dynamics. In *ICLR*, 2023. 1, 2, 4