Drawing Early-Bird Tickets: Towards More Efficient Training of Deep Networks

ICLR 2020
Citation count: 8

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Richard G. Baraniuk & Yingyan Lin*
Xiaohan Chen & Zhangyang Wang*

Motivation

- Lottery Hypothesis claim that exist a smaller and outperform network
- Iterative pruning costs time
- It seems that lottery ticket fail while large learning rate

Background

- Lottery Hypothesis
- The limit of lottery hypothesis

Background

- Lottery Hypothesis

  A neural network θ with initial parameters θ = $\theta_0 \sim D_\theta$. When optimizing with SGD , it reaches minimum validation loss l at iteration j with test accuracy a . In addition, consider training $m \odot \theta_0$ with a mask m ∈ {0, 1} and optimizing with SGD on the same training set, it reaches minimum validation loss l’ at iteration j’ with test accuracy a’.

  The lottery ticket hypothesis predicts that **∃ m for which j’ 0 ≤ j, a’ ≥ a , and $\|m\|_2$ < |θ| (fewer parameters).**

Background

- Random Initialize

With randomly reinitialization ($m \odot \theta_0$ where θ = $\theta_0 \sim D_\theta$) winning tickets **no longer match the performance of the original network**

- However, refer to ***Rethinking the value of network pruning(Liu et al. ICLR 2019),*** they seemingly find a result contradict with it. I will show it later.

Problem formulation

- Is there a more efficient training framework to get the lottery?

Early-Bird Ticket Algorithm

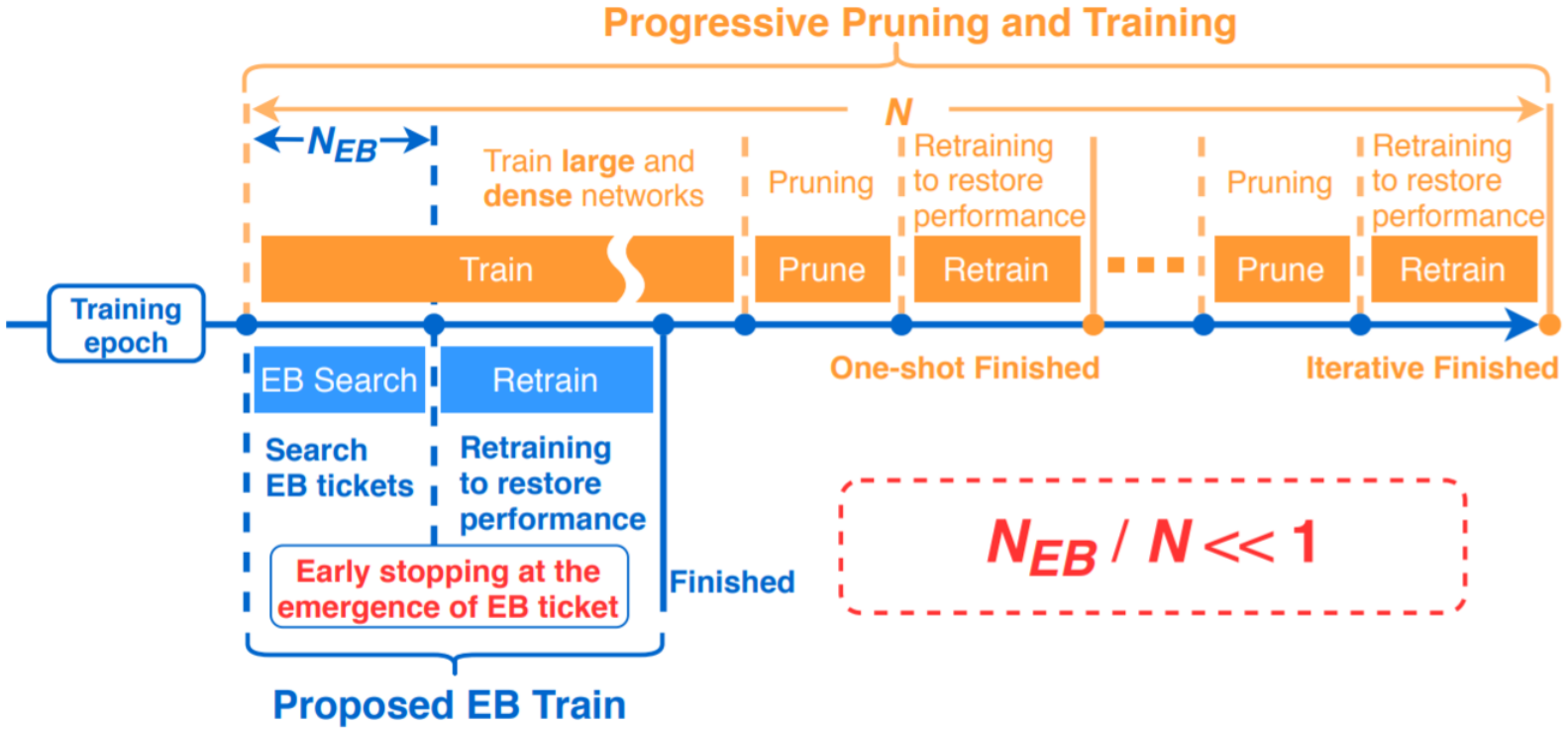

Figure 4: A high-level overview of the difference between EB Train and existing *progressive pruning and training*.

N_{EB}/N = 12.5%, 6.25% in the experiments (Figure1, Table 1)

Propose a new method to measure pruned network(mask * θ), s.t. the ticket search can be early stopped

Outline

- Main Idea
- Validation
- Efficient Early-Bird Training & Experiments
- Comparing with Lottery
- Conclusion

Main Idea

- **Early-Bird Ticket Hypothesis**

  Consider a dense, randomly-initialized network f(x; θ), f reaches a minimum validation loss floss at the i-th iteration with a test accuracy f_{acc}, when optimized with SGD on a training set.

  In addition, consider subnetworks f(x; m ⊙ θ) with a mask m ∈ {0, 1}. When being optimized with SGD on the same training set, f(x; m ⊙ θ) reach loss f′ loss at the i′-th iteration with a test accuracy f'_{acc}.

  Hypothesis claim that **exists m such that $f'_{acc} \approx f_{acc}$ (even ≥), i.e., same or better generalization, with i′ ≪ i and sparse m** (i.e., much reduced parameters).

Main Idea

- **Implement Early-Bird Ticket**

Algorithm 1: EB Ticket Searching Algorithm

Initialize the weights $\boldsymbol{W}$, scaling factor r, pruning ratio p, and the FIFO queue Q with length l;
while t (epoch) $< t_{max}$ **do**
    Update $\boldsymbol{W}$ and r using SGD training;
    Perform structured pruning based on r_t towards the target ratio p;
    Calculate the **mask distance** between the current and last subnetworks and add to Q.
    $t = t + 1$
    if Max$(Q) < \epsilon$ **then**
        $t^* = t$
        Return $f(x; m_{t^*} \odot \boldsymbol{W})$ (EB ticket);
    end if
end while

The pruning ratio p, length of queue and threshold Ɛ are hyperparameters
By default Ɛ = 0.1, l = 5

Main Idea

- Measure **mask distance by Hamming Distance**
- Which measuring the **difference** between 2 masks
- Examples
  - "**karolin**" and "**kathrin**" is 3.
  - "**karolin**" and "**kerstin**" is 3.
  - "**kathrin**" and "**kerstin**" is 4.
  - **1011101** and **1001001** is 2.
  - **2173896** and **2233796** is 3.

Validation

- Visualize the distance of masks drawn from each epoch

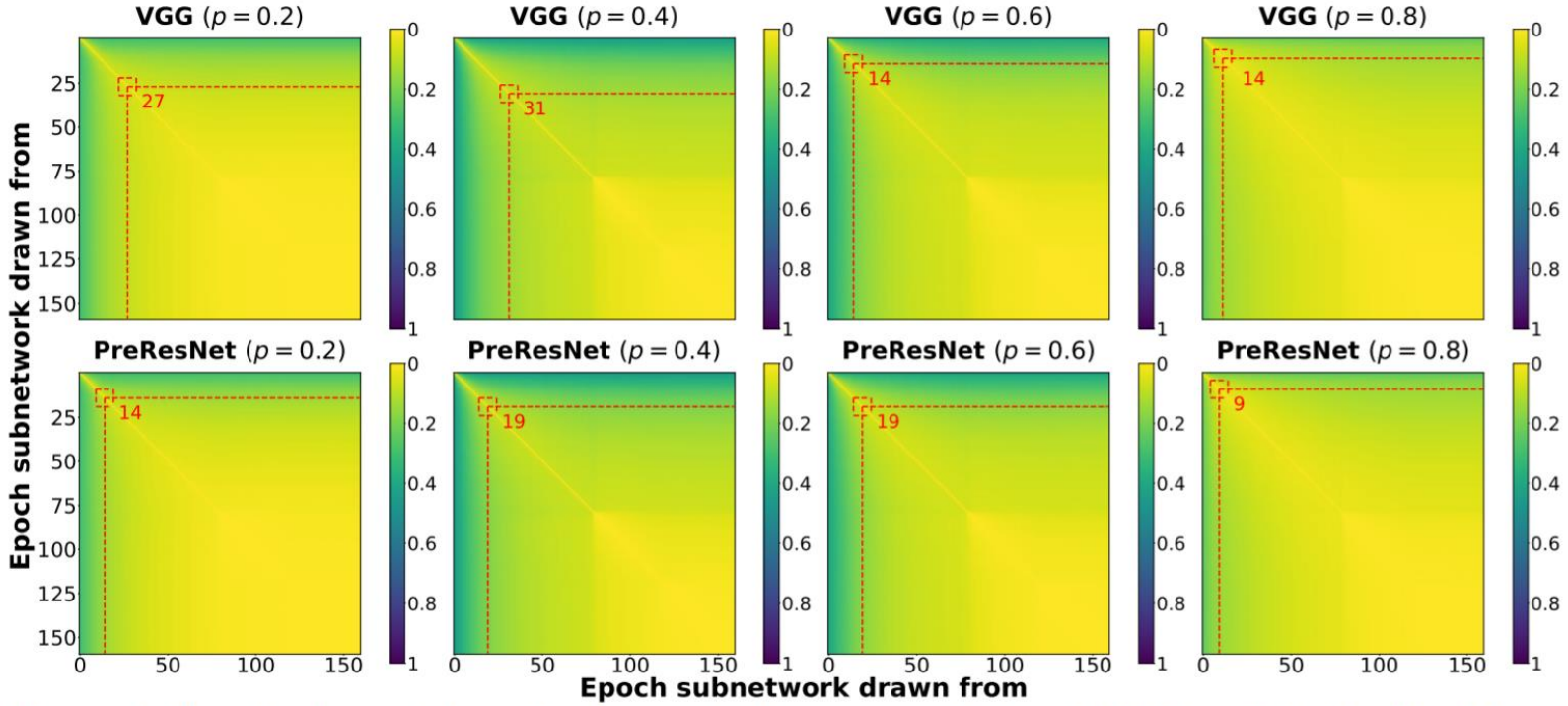

Figure 3: Visualization of the pairwise mask distance matrix for VGG16 and PreResNet101 on CIFAR-100.

X, Y axis are epochs that masks are drawn from. The **red lines denote 0.1 in mask distance**

Validation

- EB Tickets Emerge & Performance

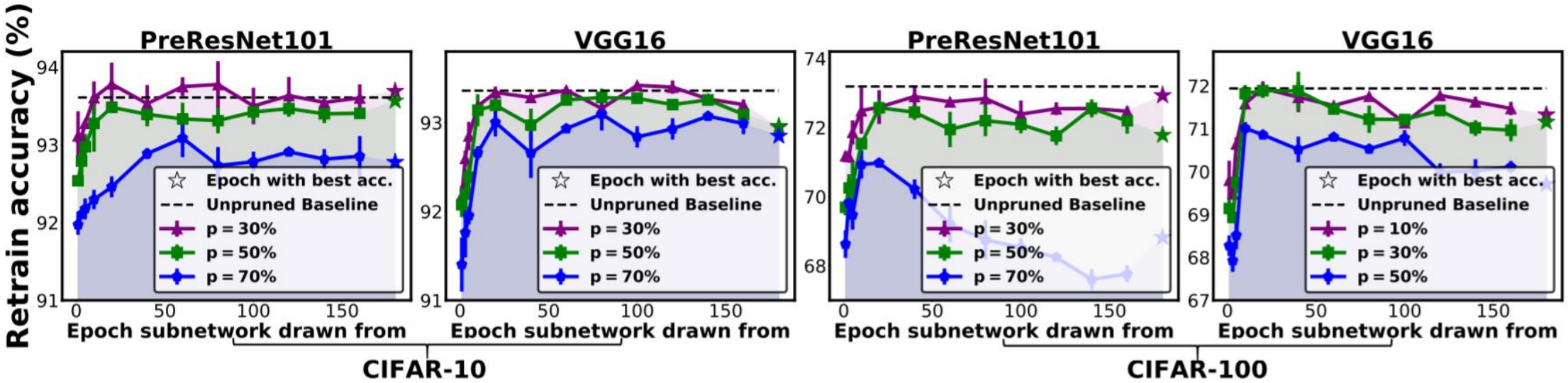

Figure 1: Retraining accuracy vs. epoch number at which the subnetworks are drawn, for both PreResNet101 and VGG16 on CIFAR-10/100 datasets, where p indicates the channel pruning ratio and the dashed line shows the accuracy of the corresponding dense model on the same dataset, ☆ denotes the retraining accuracies of subnetworks drawn from the epochs with the best searching accuracies. Error bars are the minimum and maximum of three runs.

N_{EB} /N = 12.5% -> 19-th epoch

Validation

- Learning Rate & Performance

Table 1: The retraining accuracy of subnetworks drawn at different training epochs using different learning rate schedules, with a pruning ratio of 0.5. Here [0, 100] represents $[0_{LR\to 0.01}, 100_{LR\to 0.001}]$ while [80, 120] denotes $[80_{LR\to 0.01}, 120_{LR\to 0.001}]$, for compactness.

	LR Schedule	Retrain acc. (%) (CIFAR-100)				Retrain acc. (%) (CIFAR-10)			
		10	20	40	final	10	20	40	final
VGG16	[0, 100]	66.70	67.15	66.96	69.72	92.88	93.03	92.80	92.64
	[80, 120]	**71.11**	**71.07**	**69.14**	**69.74**	**93.26**	**93.34**	**93.20**	**92.96**
PreResNet101	[0, 100]	69.68	69.69	69.79	70.96	92.41	92.72	92.42	93.05
	[80, 120]	**71.58**	**72.67**	**72.67**	**71.52**	**93.60**	**93.46**	**93.56**	**93.42**

Pruning Rate

N_{EB} /N = 6.25% -> 10-th epoch

[80, 120] means that starting from 0.1, decay to 0.01 at 80-th epoch and further decay to 0.001 at 120 epoch

Validation

- Learning Rate & Performance

Table 4: The retraining accuracy of subnetworks drawn at different training epochs using different inital learning rate, where the pruning ratio is 0.5.

	LR Initial	Retrain acc.(%) (CIFAR-100)				Retrain acc.(%) (CIFAR-10)			
		10	20	40	final	10	20	40	final
VGG16	0.1	**71.11**	71.07	69.14	69.74	93.26	93.34	93.20	92.96
	0.5	70.69	**71.65**	**71.94**	**71.58**	**93.49**	**93.45**	**93.44**	**93.29**
PreResNet101	0.1	71.58	72.67	72.67	71.52	**93.60**	93.46	93.56	93.42
	0.2	**72.58**	**72.90**	**72.86**	**72.71**	93.40	**93.46**	**93.87**	**93.69**

Pruning Rate

0.5 means that starting from 0.5, decay to 0.01 at 80-th epoch
and further decay to 0.001 at 120 epoch

Validation

- Precision & Performance & Total Training FLOPs

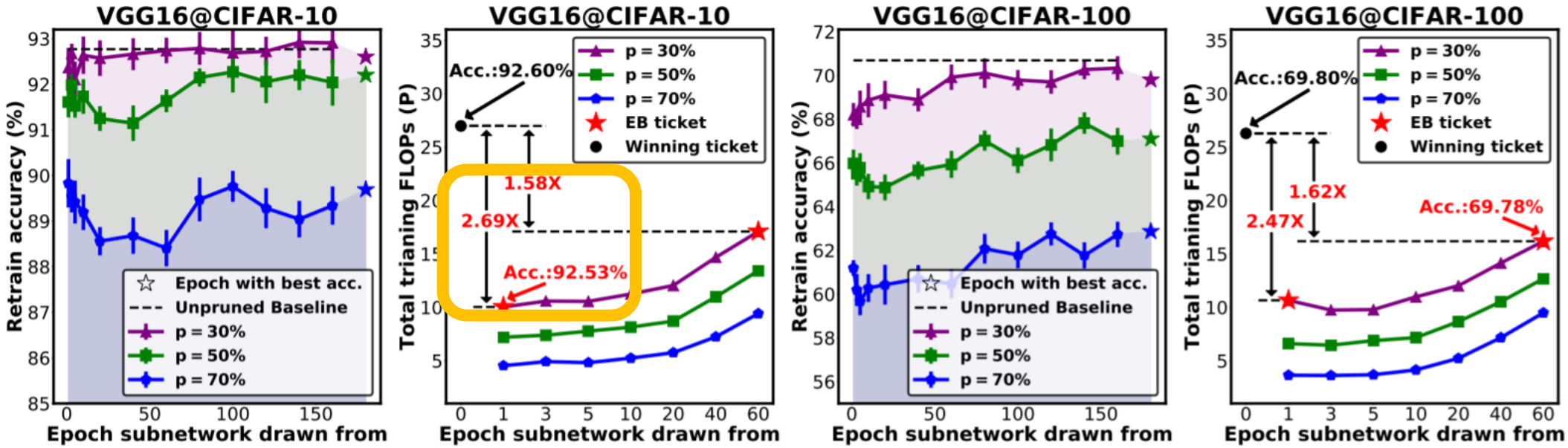

Figure 2: Retraining accuracy and total training FLOPs comparison vs. epoch number at which the subnetwork is drawn, when using 8-bit precision during the stage of identifying EB tickets based on the VGG16 model and CIFAR-10/100 datasets, where p indicates the channel pruning ratio and the dashed line shows the accuracy of the corresponding dense model on the same dataset.

Ticket search in 8 bits + Retrain in 32 bits full-precision

Validation

- Precision & Performance & Total Training FLOPs

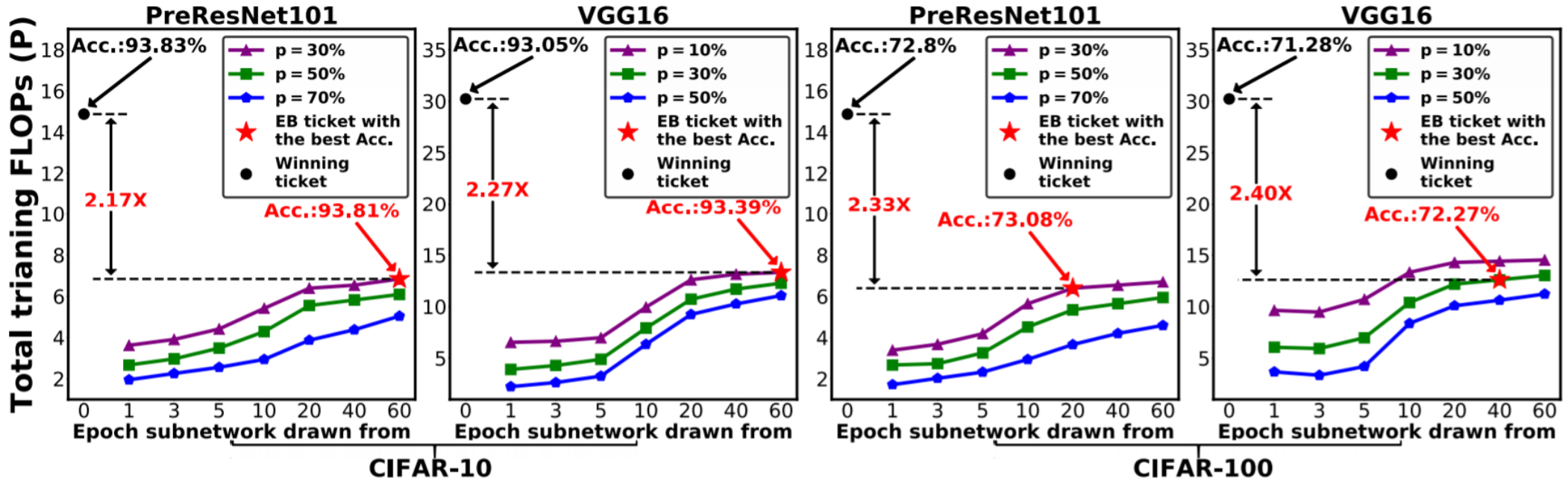

Figure 5: The total training FLOPs vs. the epochs at which the subnetworks are drawn from, for both the PreResNet101 and VGG16 models on the CIFAR-10 and CIFAR-100 datasets, where p indicates the channel-wise pruning ratio for extracting the subnetworks. Note that the EB tickets at all cases achieve higher accuracies and consume less computational FLOPs than those of the "ground-truth" winning tickets (drawn after the full training of 160 epochs).

Ticket search in 8 bits + Retrain in 32 bits full-precision

Experiments

	Ticket Search	Retrain
LF	8 bits	32 bits
FF	32 bits	32 bits
Re-init	Retrain from a new random initialization	

Table 2: Comparing the accuracy and energy/FLOPs of EB Train (including its variants), NS (Liu et al. (2017)), LT (Frankle & Carbin, 2019), SNIP (Lee et al., 2019) and ThiNet (Luo et al. (2017)).

Setting	Methods	Retrain acc.			Energy cost (KJ)/FLOPs (x0.05P)		
		p=30%	p=50%	p=70%	p=30%	p=50%	p=70%
PreResNet -101 CIFAR-10	LT (one-shot)	93.70	93.21	**92.78**	6322/298	6322/298	6322/298
	SNIP	93.76	93.31	92.76	3161/148	3161/148	3161/148
	NS	93.83	93.42	92.49	5270/278	4641/254	4211/220
	ThiNet	93.39	93.07	91.42	3579/264	2656/211	1901/173
	EB Train (re-init)	93.88	93.29	92.39	2817/155	2382/141	1565/75.4
	EB Train	**93.91**	**93.90**	92.49	2370/130	1970/114	1452/69.9
	EB Train (LF)	93.48	93.31	92.24	2265/130	1667/102	1338/64.7
	EB Train (LL)	93.24	92.85	92.12	**489.4/130**	**410.9/102**	**281.8/64.7**
	EB Train Improv.	**0.08**	**0.48**	-0.29	**6.5×/1.1×**	**6.5×/1.5×**	**6.7×/2.3×**
VGG16 CIFAR-10	LT (one-shot)	93.18	93.25	**93.28**	746.2/605	746.2/605	746.2/605
	SNIP	93.20	92.71	92.30	373.1/302	373.1/302	373.1/302
	NS	93.05	92.96	92.70	617.1/547	590.7/514	553.8/475
	ThiNet	92.82	91.92	90.40	298.0/451	383.9/379	380.1/331
	EB Train (re-init)	93.11	93.23	92.71	290.4/288	237.3/240	200.5/189
	EB Train	**93.39**	**93.26**	92.71	256.4/**253**	213.4/215	184.2/197
	EB Train (LF)	93.20	93.19	92.91	250.1/261	199.4/214	170.3/164
	EB Train (LL)	93.25	93.13	92.60	**56.1**/261	**43.1/214**	**36.5/164**
	EB Train Improv.	**0.19**	**0.01**	- 0.57	**6.6×/1.2×**	**8.6×/1.4×**	**10.2×/1.8×**
PreResNet -101 CIFAR-100	LT (one-shot)	71.90	71.60	69.95	6095/298	6095/298	6095/298
	SNIP	72.34	71.63	70.01	3047/148	3047/148	3047/148
	NS	72.80	71.52	68.46	4851/274	4310/250	3993/206
	ThiNet	73.10	70.92	67.29	3603/264	2642/211	1893/173
	EB Train (re-init)	73.23	**73.36**	71.05	2413/147	2016/125	1392/70.6
	EB Train	**73.52**	73.15	**72.29**	2020/128	1769/109	1294/65.6
	EB Train (LF)	73.41	73.02	70.72	2038/129	1614/105	1171/59.8
	EB Train (LL)	73.04	71.82	69.45	**434.4/129**	**366.5/105**	**247.3/59.8**
	EB Train Improv.	**0.42**	**1.73**	**2.28**	**7.0×/1.2×**	**7.2×/1.4×**	**7.6×/2.5×**
		p=10%	p=30%	p=50%	p=10%	p=30%	p=50%
VGG16 CIFAR-100	LT (one-shot)	**72.62**	71.31	70.96	741.2/605	741.2/605	741.2/605
	SNIP	71.55	70.83	70.35	370.6/302	370.6/302	370.6/302
	NS	71.24	71.28	69.74	636.5/586	592.3/542	567.8/480
	ThiNet	70.83	69.57	67.22	632.2/547	568.5/451	381.4/379
	EB Train (re-init)	71.65	71.48	69.66	345.3/326	300.0/273	246.8/212
	EB Train	71.81	**72.17**	**71.28**	287.7/282	262.2/**243**	221.7/197
	EB Train (LF)	71.60	71.50	70.27	270.5/277	262.7/252	208.7/194
	EB Train (LL)	71.34	70.53	69.91	**54.6/277**	**64.4**/252	**44.4/194**
	EB Train Improv.	- 0.81	**0.86**	**0.32**	**6.8×/1.1×**	**5.8×/1.2×**	**10.7×/1.6×**

Conclusion

- **The pruning result has been determined in very early stage**
- The precision of weight doesn’t matter
- It seems that retraining **from random initialization doesn’t hurt the performance of EB ticket**
- The learning rate used here is **much larger** than lottery tickets

Learning Rate Matters

Rethinking the value of network pruning

ICLR 2019
Citation count: 263

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, Trevor Darrell

Structured Pruning & Unstructured Pruning

- **Structured Pruning:**

Predefine p% pruning rate(pruned nodes / all nodes) for each layers

- **Unstructured Pruning:**

Prune with different pruning rate(for each layers automatically

(Lottery Ticket)

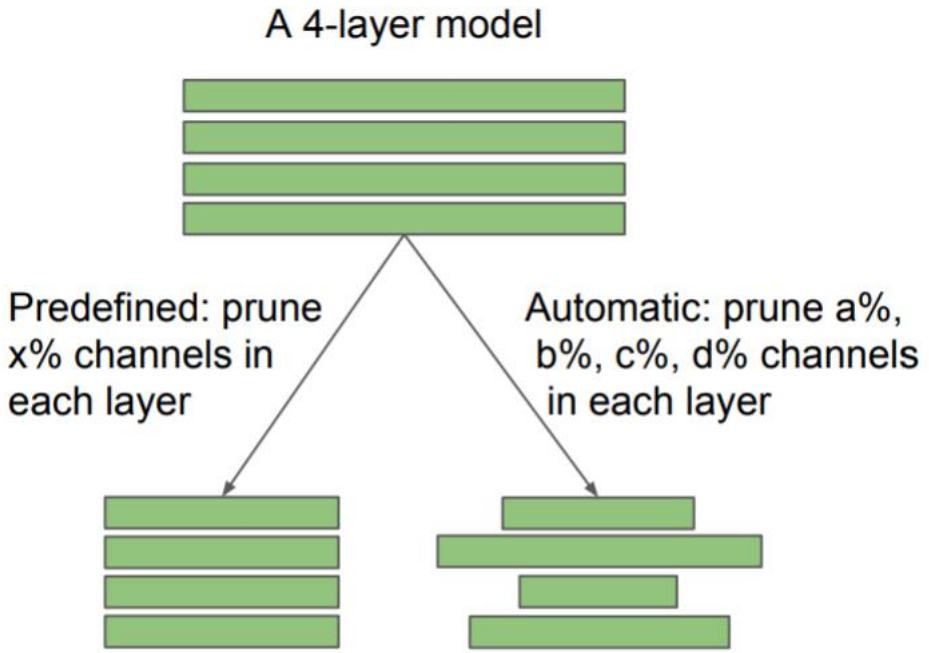

Figure 2: Difference between predefined and automatically discovered target architectures, in channel pruning as an example. The pruning ratio x is user-specified, while a, b, c, d are determined by the pruning algorithm. Unstructured sparse pruning can also be viewed as automatic.

Settings of Experiment on Lottery Ticket

- Compare the result of **structured pruning method**, despite for Lottery Ticket only evaluates on unstructured pruning.
- Relatively large modern models VGG16, ResNet50 used in the original pruning method than Lottery Ticket
- Use momentum SGD with a **large initial learning rate (0.1)**, while Lottery Ticket only use much smaller ones
- Include the large-scale **ImageNet dataset** while Lottery Ticket only considers MNIST and CIFAR

Unstructured Pruning

1. With large learning rate, performance of random initialization and lottery ticket are close

2. Using the original initialization as in Lottery Ticket only **provides advantage over random initialization with small initial learning rate 0.01**

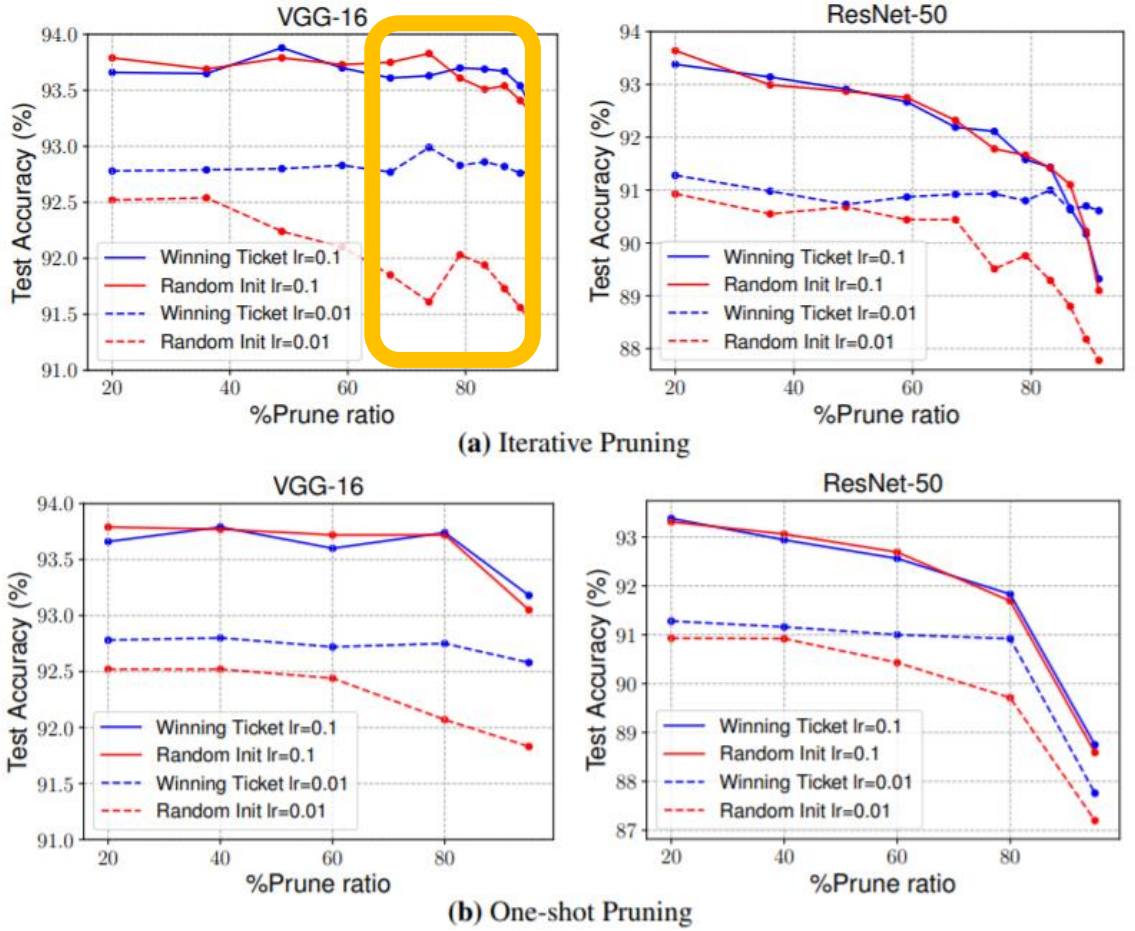

Figure 7: Comparisons with the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) for iterative/one-shot unstructured pruning (Han et al., 2015) with two initial learning rates 0.1 and 0.01, on CIFAR-10 dataset. Each point is averaged over 5 runs. Using the winning ticket as initialization only brings improvement when the learning rate is small (0.01), however such small learning rate leads to a lower accuracy than the widely used large learning rate (0.1).

Unstructured Pruning

1. With large learning rate, performance of random initialization and lottery ticket are close

2. Using the original initialization as in Lottery Ticket only **provides advantage over random initialization with small initial learning rate 0.01**

Dataset	Model	Unpruned	Prune Ratio	Winning Ticket	Random Init
CIFAR-10	VGG-16	93.76 (±0.20)	20%	93.66 (±0.20)	**93.79** (±0.11)
			40%	**93.79** (±0.12)	93.77 (±0.10)
			60%	93.60 (±0.13)	**93.72** (±0.11)
			80%	**93.74** (±0.15)	93.72 (±0.16)
			95%	**93.18** (±0.12)	93.05 (±0.21)
	ResNet-50	93.48 (±0.20)	20%	**93.38** (±0.18)	93.31 (±0.24)
			40%	92.94 (±0.12)	**93.06** (±0.22)
			60%	92.56 (±0.20)	**92.69** (±0.11)
			80%	**91.83** (±0.20)	91.69 (±0.21)
			95%	**88.75** (±0.18)	88.59 (±0.09)

(a) One-shot pruning with initial learning rate 0.1

Dataset	Model	Unpruned	Prune Ratio	Winning Ticket	Random Init
CIFAR-10	VGG-16	92.69 (±0.12)	20%	**92.78** (±0.11)	92.52 (±0.15)
			40%	**92.80** (±0.18)	92.52 (±0.15)
			60%	**92.72** (±0.16)	92.44 (±0.19)
			80%	**92.75** (±0.07)	92.07 (±0.25)
			95%	**92.58** (±0.25)	91.83 (±0.11)
	ResNet-50	91.06 (±0.28)	20%	**91.28** (±0.15)	90.93 (±0.34)
			40%	**91.16** (±0.07)	90.92 (±0.10)
			60%	**91.00** (±0.15)	90.43 (±0.16)
			80%	**90.92** (±0.08)	89.71 (±0.18)
			95%	**87.76** (±0.19)	87.20 (±0.17)

(b) One-shot pruning with initial learning rate 0.01

Table 9: Comparisons with the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) for one-shot unstructured pruning (Han et al., 2015) with two initial learning rates: 0.1 and 0.01. The same results are visualized in Figure 7b. Using the winning ticket as initialization only brings improvement when the learning rate is small (0.01), however such small learning rate leads to a lower accuracy than the widely used large learning rate (0.1).

Structured Pruning

For structured pruning, Lottery Ticket **doesn't outperform random initialization**

Dataset	Model	Unpruned	Pruned Model	Winning Ticket	Random Init
CIFAR-10	VGG-16	93.63 (±0.16)	VGG-16-A	**93.62** (±0.09)	93.60 (±0.15)
	ResNet-56	93.14 (±0.12)	ResNet-56-A	92.72 (±0.10)	**92.75** (±0.26)
			ResNet-56-B	92.78 (±0.23)	**92.90** (±0.27)
	ResNet-110	93.14 (±0.24)	ResNet-110-A	**93.21** (±0.09)	**93.21** (±0.21)
			ResNet-110-B	93.15 (±0.12)	**93.37** (±0.29)

(a) Initial learning rate 0.1

Dataset	Model	Unpruned	Pruned Model	Winning Ticket	Random Init
CIFAR-10	VGG-16	92.64 (±0.05)	VGG-16-A	92.65 (±0.18)	**92.67** (±0.22)
	ResNet-56	89.81 (±0.27)	ResNet-56-A	**90.00** (±0.15)	89.87 (±0.25)
			ResNet-56-B	89.75 (±0.35)	**89.81** (±0.24)
	ResNet-110	89.43 (±0.39)	ResNet-110-A	89.48 (±0.35)	**89.49** (±0.10)
			ResNet-110-B	**89.36** (±0.30)	89.35 (±0.16)

(b) Initial learning rate 0.01

Table 8: Comparisons with the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) on a structured pruning method (L_1-norm based filter pruning (Li et al., 2017)) with two initial learning rates: 0.1 and 0.01. In both cases, using winning tickets does not bring improvement on accuracy.

Confirm by Lottery Hypothesis

1. Doesn’t outperform but the pattern still emerge
2. Training VGG-19 with warmup (k = 10000) improves the test accuracy

Learning Rate 0.1

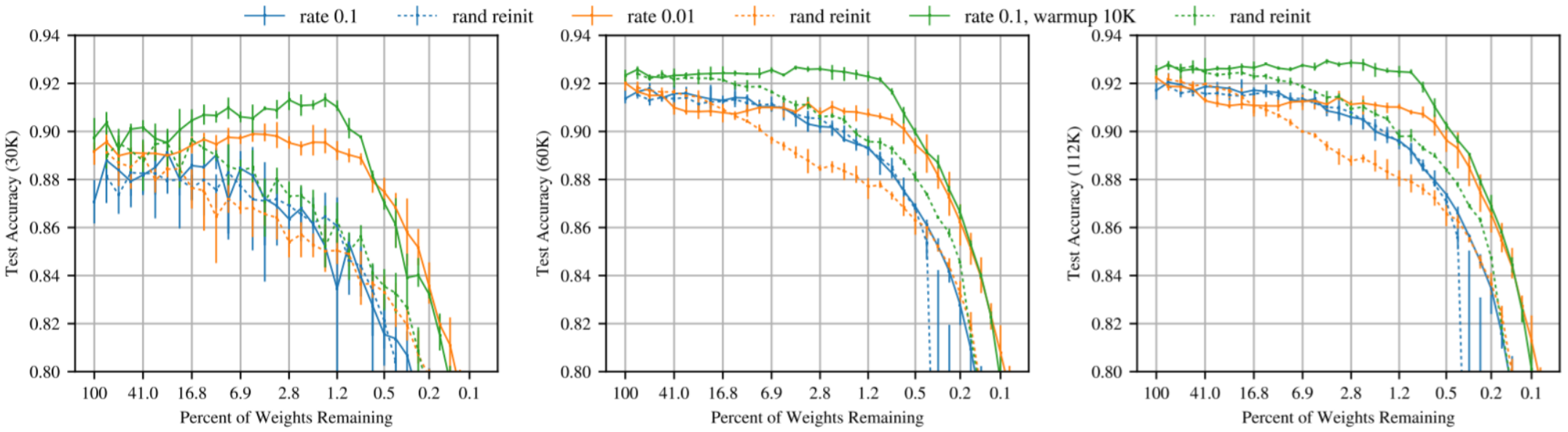

Figure 7: Test accuracy (at 30K, 60K, and 112K iterations) of VGG-19 when iteratively pruned.

Confirm by Lottery Hypothesis

Hypothesis

- Highly overparameterized networks can be pruned, reinitialized, and retrained successfully
- Less severely overparamterized networks only maintain accuracy with fortuitous initialization.

Conclusion

- The lottery ticket **only brings improvement in the case of unstructured pruning, with small initial learning rate**, but this small learning rate yiel ds inferior accuracy compared with the widely-used large learning rate.
- For structured pruning,
  training the small target model from **random initialization can achieve t he same performance of unpruned model**

End

Background

- ***Rethinking the value of network pruning(Liu et al. ICLR 2019)***.
- *They found* the winning **ticket only brings improvement in the case of unstructured pruning, with small initial learning rate**, but this small learning rate **yields inferior accuracy compared with the widely-used large learning rate**.