Superposing Many Tickets into One: A Performance Booster for Sparse Neural Network Training Supplementary Materia

Lu Yin[1] **Vlado Menkovski**[1] **Meng Fang**[1] **Tianjin Huang**[1] **Yulong Pei**[1] **Mykola Pechenizkiy**[1]
Decebal Constantin Mocanu[2,1] **Shiwei Liu**[1]

[1]Eindhoven University of Technology , Eindhoven, the Netherlands
[2]University of Twente, Enschede, the Netherlands

A EXPERIMENTAL RESULTS OF WIDE RESNET28-10 ON CIFAR-10/100

Table 1: Test accuracy (%) of sparse Wide ResNet28-10 on CIFAR-10/100. All the results are averaged from three random runs. In each setting, the best results are marked in bold.

Dataset	CIFAR-10			CIFAR-100		
Wide ResNet28-10 (Dense)	96.00±0.13	-	-	81.09±0.19	-	-
Sparsity	95%	90%	80%	95%	90%	80%
SET [Mocanu et al., 2018]	**95.63±0.08**	95.85±0.02	95.92±0.25	79.36±0.14	80.44±0.18	80.60±0.07
SET+Sup-tickets (ours)	95.53±0.11	**95.91±0.14**	**95.93±0.10**	**79.66±0.18**	**80.65±0.04**	**80.91±0.20**
RigL [Evci et al., 2020]	95.70±0.07	95.96±0.12	96.12±0.05	79.41±0.24	80.45±0.45	80.92±0.20
RigL+Sup-tickets (ours)	**95.90±0.11**	**95.98±0.06**	**96.15±0.08**	**80.00±0.15**	**80.72±0.22**	**81.16±0.09**
GraNet [**?**]	95.95±0.08	96.02±0.01	**96.09±0.07**	80.43±0.17	80.97±0.16	81.31±0.09
GraNet+Sup-tickets (ours)	**96.03±0.11**	**96.13±0.07**	96.08±0.04	**80.65±0.06**	**81.20±0.09**	**81.42±0.18**

B IMPACT OF THE CHEAP TICKETS WITHOUT TRAINING TIME CONSTRAINT

We extend the overall training time to yield 9 tickets. All the cheap tickets have been trained for 8 epochs. The results on CIFAR-100 are reported below. All results are averaged from 3 random runs. As shown, the performance of Sup-tickets continuously improves as the number of tickets increases.

Table 2: Test accuracy (%) on CIFAR-100 of Sup-tickets combined with RigL under different cheap ticket count. The best results are marked in bold.

Ticket	Sparsity		
count	95%	90%	80%
	VGG-16		
N=3	71.47±0.29	72.86±0.22	73.42±0.21
N=6	71.79±0.10	73.19±0.23	73.69±0.36
N=9	**71.92±0.07**	**73.30±0.26**	**74.00±0.38**
	ResNet-50		
N=3	77.14±0.57	77.84±0.21	78.08±0.40
N=6	77.53±0.55	78.12±0.32	78.18±0.49
N=9	**77.57±0.55**	**78.15±0.20**	**78.19±0.46**

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

C THE VARIANCE OF THE MULTIPLE CHEAP TICKETS

The variance of the cheap tickets obtained by our method is quite low, as shown in the following table. To ensure good final performance, we expect all the subnetworks to be located in the same low-loss basin with similar performance. On the other hand, high variance means that cheap tickets are located in different basins, and weight averaging will not bring performance gains. To verify this hypothesis, we generate 3 cheap subnetworks under 95% sparsity on CIFAR-100 with high variance by using different prune/grow criteria: prune with high magnitude and grow with high gradient, prune with low magnitude and grow randomly, prune with low magnitude and grow randomly.The results are reported in Table 3.

We find that averaging subnetworks with high variance significantly hurt the performance, likely due to the fact that they are not from the same loss basin.

Table 3: Accuracy (%) of each ticket and the averaged ticket under different variance.

Model	Setting	Accuracy of Each Ticket	Variance	Averaged Accuracy
ResNet-50	High Variance	[69.44, 76.52, 61.50]	6.13	2.04
	Low Variance (ours)	[77.15, 77.56, 77.07]	0.21	77.87
VGG-16	High Variance	[64.02, 70.01, 58.76]	4.60	2.14
	Low Variance (ours)	[70.31 ,70.15, 70.32]	0.08	71.19

D COMPARISON WITH LARGE LEARNING RATE SCHEDULE

We perform experiments in which the learning rate will immediately increase to a very large value of 0.1 at the beginning of each cycle. We expect that the large learning rate will force the cheap tickets to jump out of the current basin, and the weight averaging does not bring any performance gains. The results in Table 4 are perfectly in line with our expectations. Parameter averaging significantly degrades the accuracy to $10\% \sim 30\%$, even though the accuracy of each subnetwork is still high. Besides, if we generate tickets with different prune/grow criteria, they are also likely located in the different basins and dramatically hurt the performance.

Table 4: Results of Sup-ticket under large restarting learning rate (0.1) and small restarting learning rate (0.005).

Model	Setting	Accuracy of Each Ticket	Averaged Accuracy
ResNet-50	Large LR schedule	[76.05, 76.37, 76.97]	10.37
	Low LR schedule (ours)	[77.15, 77.56, 77.07]	77.87
VGG-16	Large LR schedule	[68.68, 69.29, 70.36]	31.74
	Low LR schedule (ours)	[70.31 ,70.15, 70.32]	71.19

E IMPLEMENTATION DETAILS OF SUP-TICKETS

In this appendix, we report the implementation details for Sup-tickets, including: total training epochs (T-epochs), epochs of normal sparse training (N-epochs), epochs of cheap tickets generation (C-epochs), length of per cyclical learning rate schedule (C), learning rate (LR), batch size (BS), learning rate drop (LR Drop), the lowest learning rate of cyclical learning rate schedule (LR-α_1), the largest learning rate of cyclical learning rate schedule (LR-α_2), weight decay (WD), produced tickets count (Ticket Count), SGD momentum (Momentum), sparse initialization (Sparse Init), etc.

E.1 IMPLEMENTATION DETAILS FOR CIFAR-10/100

Table 5: Implementation hyperparameters of Sup-tickets on CIFAR-10/100

Model	T-epochs	N-epochs	C-epochs	C	BS	LR	LR Drop, Epochs	LR-α_2	LR-α_1	Ticket Count	Optimizer	WD	Momentum	Sparse Init
VGG-16	250	226	24	8	128	0.1	10x, [113, 169]	0.001	0.005	3	SGD	0.9	5e-4	ERK
ResNet-50	250	226	24	8	128	0.1	10x, [113, 169]	0.001	0.005	3	SGD	0.9	5e-4	ERK
Wide ResNet28-10	250	226	24	8	128	0.1	10x, [113, 169]	0.001	0.005	3	SGD	0.9	5e-4	ERK

E.2 IMPLEMENTATION DETAILS FOR IMAGENET

Table 6: Implementation hyperparameters of Sup-tickets on ImageNet

Model	T-epochs	N-epochs	C-epochs	C	BS	LR	LR Drop, Epochs	LR-α_2	LR-α_1	Ticket Count	Optimizer	WD	Momentum	Sparse Init
ResNet-50	100	92	8	2	64	0.1	10x, [30, 60, 85]	0.0001	0.0005	4	SGD	0.9	1e-4	ERK

F COMPARISON BETWEEN DIFFERENT BATCH NORMALIZATION UPDATING STRATEGIES.

In this section, we compare the test accuracy between two batch normalization updating strategies: (1) using additional running pass over the training data; (2) retrieving the statistic by averaging across each cheap ticket (ours). From Table 7 and Table 8, we find that there is no obvious difference in test accuracy between these two methods. However, our method could save extra computation resources without the additional running pass.

Table 7: Test accuracy (%) of different batch normalization updating strategies for ResNet 50 on ImageNet. BU stands for batch normalization updating using additional running pass over the data. AV means averaging across each cheap ticket (ours). In each setting, the best results are marked in bold.

Dataset	ImageNet	
Sparsity	90%	80%
RigL+Sup-tickets (AV)	74.044	**75.966**
RigL+Sup-tickets (BU)	**74.083**	75.925
GraNet+Sup-tickets (AV)	74.554	**76.168**
GraNet+Sup-tickets (BU)	**74.560**	76.109

Table 8: Test accuracy (%) of different batch normalization updating strategies on CIFAR-10/100. BU stands for batch normalization updating using additional running pass over the data. AV means averaging across each cheap ticket (ours). In each setting, the best results are marked in bold.

Dataset	CIFAR-10			CIFAR-100		
Sparsity	95%	90%	80%	95%	90%	80%
VGG-16 (Dense)	93.91±0.26	-	-	73.61±0.45	-	-
SET+Sup-tickets (AV)	93.22±0.09	**93.63±0.05**	93.80±0.13	71.18±0.29	**71.99±0.27**	73.02±0.32
SET+Sup-tickets (BU)	93.22±0.12	93.62±0.01	93.80±0.01	**71.30±0.26**	71.96±0.19	**73.04±0.31**
RigL+Sup-tickets (AV)	93.20±0.13	93.81±0.11	93.85±0.25	71.31±0.21	72.57±0.29	73.61±0.11
RigL+Sup-tickets (BU)	**93.24±0.11**	**93.86±0.15**	**93.88±0.28**	**71.36±0.16**	**72.60±0.27**	**73.68±0.16**
GraNet+Sup-tickets (AV)	94.10±0.06	**94.13±0.12**	94.24±0.05	73.61±0.24	**73.87±0.26**	73.95±0.30
GraNet+Sup-tickets (BU)	**94.14±0.06**	94.10±0.14	**94.25±0.07**	**73.71±0.21**	73.79±0.21	**74.03±0.27**
Wide ResNet28-10 (Dense)	96.00±0.13	-	-	81.09±0.19	-	-
SET+Sup-tickets (AV)	95.53±0.11	95.91±0.14	95.92±0.10	**79.66±0.18**	**80.65±0.04**	**80.91±0.20**
SET+Sup-tickets (BU)	**95.59±0.11**	**95.98±0.08**	**95.97±0.06**	79.36±0.35	80.47±0.05	80.74±0.21
RigL+Sup-tickets (AV)	**95.90±0.11**	**95.98±0.06**	96.15±0.08	**80.00±0.15**	**80.72±0.22**	**81.16±0.09**
RigL+Sup-tickets (BU)	95.88±0.10	95.97±0.04	**96.17±0.11**	79.76±0.23	80.52±0.20	81.13±0.15
GraNet+Sup-tickets (AV)	**96.03±0.11**	96.13±0.07	96.08±0.04	80.65±0.06	**81.20±0.09**	**81.42±0.18**
GraNet+Sup-tickets (BU)	96.01±0.07	**96.19±0.08**	**96.14±0.09**	**80.73±0.04**	81.17±0.13	81.39±0.21
ResNet-50 (Dense)	94.88±0.11	-	-	78.00±0.40	-	-
SNIP+Sup-tickets (AV)	94.33±0.09	95.05±0.22	95.21±0.09	**65.56±1.15**	76.34±0.27	**77.43±0.53**
SNIP+Sup-tickets (BU)	**94.39±0.06**	**95.10±0.12**	**95.30±0.02**	65.51±0.83	**76.62±0.23**	77.35±0.62
ERK+Sup-tickets (AV)	93.92±0.04	94.80±0.06	95.11±0.27	75.75±0.28	76.82±0.08	**77.85±0.42**
ERK+Sup-tickets (BU)	**93.99±0.08**	**94.87±0.04**	**95.18±0.27**	**76.02±0.22**	**77.01±0.17**	77.80±0.54
SET+Sup-tickets (AV)	94.81±0.05	94.87±0.03	**94.90±0.27**	**76.68±0.38**	77.89±0.45	78.35±0.18
SET+Sup-tickets (BU)	**94.85±0.03**	**94.97±0.05**	94.86±0.20	76.54±0.41	**77.93±0.50**	**78.38±0.18**
RigL+Sup-tickets (AV)	**94.65±0.11**	94.82±0.13	**94.81±0.15**	**77.58±0.47**	**78.52±0.39**	**78.69±0.30**
RigL+Sup-tickets (BU)	94.64±0.13	**94.89±0.09**	94.79±0.17	77.54±0.53	78.43±0.40	78.53±0.31
GraNet+Sup-tickets (AV)	94.89±0.15	95.08±0.08	94.94±0.03	77.70±0.47	78.37±0.53	**78.95±0.33**
GraNet+Sup-tickets (BU)	**94.91±0.19**	**95.16±0.14**	**95.09±0.03**	**77.82±0.60**	**78.63±0.64**	78.07±0.32

G LAYER-WISE SPARSITY OF RESNET-50 ON IMAGENET

Table 9 summarizes the final sparsity budgets for 90% sparse ResNet-50 on ImageNet-1K obtained by various methods. Backbone represents the sparsity budgets for all the CNN layers without the last fully-connected layer.

Table 9: ResNet-50 Learnt Budgets and Backbone Sparsities at Sparsity 90%

Metric	Fully Dense Params	Fully Dense FLOPs	Sparsity (%)			
			GraNet+Sup-tickets	GraNet	RigL+Sup-tickets	RigL
Overall	25502912	8178569216	89.99	89.98	90.23	90.00
Backbone	23454912	8174272512	89.89	90.65	92.47	90.00
Layer 1 - conv1	9408	118013952	37.40	38.22	57.26	58.32
Layer 2 - layer1.0.conv1	4096	236027904	40.55	41.70	14.58	9.40
Layer 3 - layer1.0.conv2	36864	231211008	64.88	65.05	82.13	82.40
Layer 4 - layer1.0.conv3	16384	102760448	64.69	65.09	17.13	16.41
Layer 5 - layer1.0.downsample.0	16384	102760448	74.75	74.99	29.10	24.25
Layer 6 - layer1.1.conv1	16384	102760448	66.33	66.75	19.72	19.02
Layer 7 - layer1.1.conv2	36864	231211008	62.25	62.62	82.05	82.44
Layer 8 - layer1.1.conv3	16384	102760448	57.99	58.57	4.79	4.07
Layer 9 - layer1.2.conv1	16384	102760448	60.15	60.60	4.85	4.19
Layer 10 - layer1.2.conv2	36864	231211008	57.15	57.45	81.73	82.06
Layer 11 - layer1.2.conv3	16384	102760448	57.10	57.47	5.13	3.88
Layer 12 - layer2.0.conv1	32768	205520896	49.90	50.42	41.61	42.37
Layer 13 - layer2.0.conv2	147456	231211008	69.44	69.49	91.09	91.25
Layer 14 - layer2.0.conv3	65536	102760448	60.42	60.74	51.43	51.98
Layer 15 - layer2.0.downsample.0	131072	205520896	87.23	87.26	71.36	71.27
Layer 16 - layer2.1.conv1	65536	102760448	84.79	84.91	52.47	52.40
Layer 17 - layer2.1.conv2	147456	231211008	83.03	83.07	91.25	91.34
Layer 18 - layer2.1.conv3	65536	102760448	70.03	70.25	52.06	52.43
Layer 19 - layer2.2.conv1	65536	102760448	79.47	79.61	52.07	52.25
Layer 20 - layer2.2.conv2	147456	231211008	81.78	81.82	91.28	91.38
Layer 21 - layer2.2.conv3	65536	102760448	73.76	73.92	51.76	51.95
Layer 22 - layer2.3.conv1	65536	102760448	74.82	74.97	51.92	52.24
Layer 23 - layer2.3.conv2	147456	231211008	82.78	82.81	91.22	91.33
Layer 24 - layer2.3.conv3	65536	102760448	76.61	76.73	51.86	52.01
Layer 25 - layer3.0.conv1	131072	205520896	60.53	60.81	70.98	71.39
Layer 26 - layer3.0.conv2	589824	231211008	83.45	83.41	95.66	95.72
Layer 27 - layer3.0.conv3	262144	102760448	69.56	69.73	75.77	76.06
Layer 28 - layer3.0.downsample.0	524288	205520896	95.24	95.21	85.79	85.64
Layer 29 - layer3.1.conv1	262144	102760448	91.19	91.22	76.02	76.03
Layer 30 - layer3.1.conv2	589824	231211008	92.86	92.87	95.68	95.73
Layer 31 - layer3.1.conv3	262144	102760448	80.70	80.81	75.76	75.95
Layer 32 - layer3.2.conv1	262144	102760448	90.34	90.40	76.09	76.18
Layer 33 - layer3.2.conv2	589824	231211008	93.22	93.24	95.68	95.73
Layer 34 - layer3.2.conv3	262144	102760448	83.42	83.47	76.06	76.21
Layer 35 - layer3.3.conv1	262144	102760448	89.12	89.17	76.14	76.23
Layer 36 - layer3.3.conv2	589824	231211008	93.20	93.21	95.67	95.71
Layer 37 - layer3.3.conv3	262144	102760448	86.26	86.30	76.13	76.24
Layer 38 - layer3.4.conv1	262144	102760448	88.64	88.70	75.85	75.97
Layer 39 - layer3.4.conv2	589824	231211008	94.50	94.51	95.65	95.69
Layer 40 - layer3.4.conv3	262144	102760448	87.05	87.09	75.94	76.05
Layer 41 - layer3.5.conv1	262144	102760448	87.10	87.15	75.91	76.07
Layer 42 - layer3.5.conv2	589824	231211008	95.13	95.14	95.69	95.72
Layer 43 - layer3.5.conv3	262144	102760448	88.91	88.95	76.06	76.14
Layer 44 - layer4.0.conv1	524288	205520896	72.04	72.13	85.54	85.67
Layer 45 - layer4.0.conv2	2359296	231211008	93.56	93.53	97.84	97.86
Layer 46 - layer4.0.conv3	1048576	51380224	82.00	82.01	88.01	88.09
Layer 47 - layer4.0.downsample.0	2097152	205520896	99.25	99.24	92.96	92.84
Layer 48 - layer4.1.conv1	1048576	102760448	95.73	95.74	88.02	88.07
Layer 49 - layer4.1.conv2	2359296	231211008	97.39	97.39	97.86	97.87
Layer 50 - layer4.1.conv3	1048576	102760448	91.08	91.07	88.10	88.12
Layer 51 - layer4.2.conv1	1048576	205520896	87.68	87.70	87.99	88.04
Layer 52 - layer4.2.conv2	2359296	231211008	97.02	97.01	97.86	97.86
Layer 53 - layer4.2.conv3	1048576	102760448	84.54	84.50	88.07	88.07
Layer 54 - fc	2048000	4096000	82.70	82.54	92.78	92.74

H COMPARISON WITH OUTPUTS ENSEMBLE AND KNOWLEDGE DISTILLATION

This appendix compares our approach with the prediction ensemble (averaging prediction of subnetworks). For deep ensemble, we use the same procedure to generate cheap tickets as in Sup-tickets; but instead of averaging their weights and connection topology, we save all the cheap tickets in memory and average their softmax outputs at inference stage [Huang et al., 2017, Garipov et al., 2018].

The results are reported in Table 10 & Table 11. Across extensive settings, we observe that our sup-tickets could closely match the strong baseline of output averaging. Worth noting that compared with the latter, our method does not require performing multiple forward passes for prediction nor saving all the ensemble members.

Table 10: **Comparison with prediction ensemble.** Test accuracy (%) of Sup-tickets and naive deep ensemble on CIFAR10/100.

Dataset	CIFAR-10			CIFAR-100		
Sparsity	95%	90%	80%	95%	90%	80%
VGG-16						
RigL + Prediction Ensemble	**93.25±0.18**	**93.82±0.09**	**93.97±0.18**	**71.80±0.24**	**73.07±0.34**	**73.80±0.21**
RigL + Sup-tickets (ours)	93.20±0.13	93.81±0.11	93.85±0.25	71.31±0.21	72.57±0.29	73.61±0.11
ResNet-50						
RigL + Prediction Ensemble	94.64±0.12	**94.94±0.06**	**94.86±0.25**	**77.66±0.4**	**78.54±0.41**	78.67±0.25
RigL + Sup-tickets (ours)	**94.65±0.11**	94.82±0.13	94.81±0.15	77.58±0.47	78.52±0.39	**78.69±0.30**

Table 11: Test accuracy (%) of Sup-tickets and naive deep ensemble for ResNet-50 on ImageNet. In each setting, the best results are marked in bold.

Dataset	ImageNet	
Sparsity	90%	80%
RigL+Sup-tickets(Ours)	74.044	75.966
RigL+Ensemble	**74.074**	**76.022**
GraNet+Sup-tickets(Ours)	74.554	76.168
GraNet+Ensemble	**74.614**	**76.198**

Besides, we also apply knowledge distillation Hinton et al. [2015] to distill the knowledge of three sup-tickets into a sparse student model. Each soft loss from the teacher model and the hard loss from the real label have equal weight in the final loss. Compared with knowledge distillation, we do not need to save all the sub-models as teacher models and do not need an extra round of training. Below we report the test accuracy of sparse VGG-16 on CIFAR-10/100. All the results are averaged from 3 random runs. Our method achieves higher accuracy (11 out of 12 cases) than the knowledge distillation based method.

Table 12: **Comparison with knowledge distillation.** Test accuracy (%) of Sup-tickets and knowledge distillation (KD). In each setting, the best results are marked in bold.

Dataset	CIFAR-10			CIFAR-100		
Sparsity	95%	90%	80%	95%	90%	80%
SET+KD	93.13±0.06	93.56±0.16	93.53±0.10	70.73±0.18	71.79±0.42	**73.06±0.02**
SET+Sup-tickets (ours)	**93.22±0.09**	**93.63±0.05**	**93.80±0.13**	**71.18±0.29**	**71.99±0.27**	73.02±0.32
RigL+KD	92.98±0.15	93.38±0.14	93.61±0.15	70.89±0.35	72.16±0.21	72.76±0.09
RigL+Sup-tickets (ours)	**93.20±0.13**	**93.81±0.11**	**93.85±0.25**	**71.31±0.21**	**72.57±0.29**	**73.61±0.11**

I STATISTICAL SIGNIFICANCE

We analyze the statistical significance of the results obtained by Sup-tickets. To measure this, we perform Kolmogorov-Smirnov test [Berger and Zhou, 2014] (KS-test). The null hypothesis is that the two independent results/samples are drawn from the same continuous distribution. If the p-value is very small (p-value <0.05), it suggests that the difference between the two sets of results is significant, and the hypothesis is rejected. Otherwise, the obtained results are close together, and the hypothesis is true. We run the experiment on sparse VGG-16, CIFAR-10/100 for 15 runs with different random seeds and report the mean accuracy, P-value, and decision of significance below.

Table 13: **Statistical Significance Analysis.**

Dataset	CIFAR-10			CIFAR-100		
Sparsity	95%	90%	80%	95%	90%	80%
SET	92.99 ±0.16	93.41±0.20	93.65±0.15	70.50±0.31	71.55±0.38	72.76±0.21
SET+Sup-tickets (ours)	**93.17±0.16**	**93.65±0.15**	**93.91±0.20**	**71.18±0.27**	**72.21±0.29**	**73.38±0.29**
P-value	5.90e-2	1.87e-2	1.02e-2	1.88e-05	1.02e-2	1.87e-2
Statistically significant	No	Yes	Yes	Yes	Yes	Yes
RigL	92.94±0.20	93.41±0.14	93.56±0.10	70.74±0.33	71.97±0.32	72.76±0.33
RigL+Sup-tickets (ours)	**93.35±0.18**	**93.69±0.08**	**93.85±0.12**	**71.41±0.29**	**72.63±0.23**	**73.26±0.29**
P-value	1.63e-4	1.88e-05	1.88e-05	1.02e-3	1.4e-06	1.87e-2
Statistically significant	Yes	Yes	Yes	Yes	Yes	Yes

J COMPARISON WITH SWA

Compared with SWA Izmailov et al. [2018], our approach provides two advantages. First of all, our method is much more training efficient as it only requires training a subset of the network during the whole training process. On the contrary, SWA requires to fully train a dense network even if it can be pruned afterward. Second, our method can efficiently discover and average *multiple sparse sub-networks with different connectivity*, whereas SWA can only average sparse subnetworks with the same sparse connectivity. Different from dense neural networks where the connectivities are fixed, numerous sparse sub-networks with different connectivities are existing for sparse training, and all of them are capable of good performance. Instead of averaging sparse neural networks with the same sparse connectivity, it is more beneficial to average multiple sparse sub-networks with different connectivities since the sparse connectivity at initialization is insufficient to guarantee good performance.

Following we compare our method with two SWA-based methods. First, we run SWA with an additional step of pruning before the averaging. Unfortunately, it conflicts with the goal of sparse training, leading to more training FLOPs. In contrast, our approach follows a sparse-to-sparse paradigm that just trains a fraction of the parameters during the whole training process. Second, we train a sparse model from scratch without considering connection exploration. The results below have empirically evaluated the benefits of our method that achieves better performance while requiring much fewer training FLOPs.

Table 14: **Comparison with SWA.** Test accuracy (%) and training FLOPs of ResNet-50 on CIFAR100. The training FLOPs are normalized with the dense model. SWA baseline[1] means we train a dense model until the first averaging operation, prune it to the target sparsity with magnitude pruning, and then run SWA without exploring sparse connectivity. SWA baseline[2] indicates we initialize a model to certain sparse levels and perform SWA without connection exploration.

Method	Accuracy			Training FLOPs ($\times 9.74e18$)		
	95% Sparsity	90% Sparsity	80% Sparsity	95% Sparsity	90% Sparsity	80% Sparsity
SWA baseline[1]	76.64±0.45	77.23±0.44	77.72±0.29	0.91×	0.92×	0.93×
SWA baseline[2]	75.66±0.45	76.67±0.14	77.50±0.36	0.11×	0.18×	0.30×
Sup-tickets (ours)	**77.58±0.47**	**78.52±0.39**	**78.69±0.30**	0.11×	0.18×	0.30×

References

Vance W Berger and YanYan Zhou. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*, 2014.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8803–8812, 2018.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *arXiv:1707.04780. Nature communications.*, 9(1):2383, 2018.