Fast as CHITA: Neural Network Pruning with Combinatorial Optimization

Group lecture opt-ml

18/09/2023

[1] Benbaki, Riade, et al. "Fast as CHITA: Neural Network Pruning with Combinatorial Optimization." arXiv preprint arXiv:2302.14623 (2023)

Motivation

Prune a ”heavy” neural network and significantly reduce the parameters with minimal loss.

Motivation

Prune a ”heavy” neural network and significantly reduce the parameters with minimal loss.

Two main approaches in literature :

- magnitude-based: use the absolute value of weight to determine its importance
- impact-based: remove weights based on how much their removal would impact the loss function

CBS (Combinatorial Brain Surgeon) [2] is an optimization-based approach that considers the *joint effect* of multiple weights, but computationally expensive (Hessian of the loss function).

Motivation

Prune a ”heavy” neural network and significantly reduce the parameters with minimal loss.

Two main approaches in literature :

- magnitude-based: use the absolute value of weight to determine its importance
- impact-based: remove weights based on how much their removal would impact the loss function

CBS (Combinatorial Brain Surgeon) [2] is an optimization-based approach that considers the *joint effect* of multiple weights, but computationally expensive (Hessian of the loss function).

CHITA (Combinatorial Hessian free Iterative Thresholding Algorithm)

- consider a local quadratic approximation of the loss function
- propose an equivalent reformulation of the problem as an l_0-constrained sparse linear regression problem
- multi-stage algorithm that updates the local quadratic model during pruning, to leave the small neighborhood of the current solution

Outline

Problem setup

Empirical loss function $\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N} l_i(w)$, $w \in \mathbb{R}^p$ the vector of trainable parameters, N the number of data points.

Problem setup

Empirical loss function $\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N} l_i(w)$, $w \in \mathbb{R}^p$ the vector of trainable parameters, N the number of data points.

Given a pre-trained weight vector $\bar{w} \in \mathbb{R}^p$ and a target level of sparsity $\tau \in (0,1)$, construct a new vector w

- The loss function at w is as close as possible to the loss before pruning: $\mathcal{L}(w) \approx \mathcal{L}(\bar{w})$
- The number of nonzero weights at w respects the sparsity budget: $||w||_0 \leq (1-\tau)p$

local quadratic approximation of $\mathcal{L}$ around the pre-trained weight $\bar{w}$

$$\mathcal{L}(w) = \mathcal{L}(\bar{w}) + \nabla\mathcal{L}(\bar{w})^T(w-\bar{w}) + \frac{1}{2}(w-\bar{w})^T\nabla^2\mathcal{L}(\bar{w})(w-\bar{w}) + O(||w-\bar{w}||^3)$$

$g \approx \nabla\mathcal{L}(\bar{w})$, $H \approx \nabla^2\mathcal{L}(\bar{w})$ gradient and Hessian approximations and ignore higher-order terms, $\mathcal{L}$ can be locally approximated by:

$$\mathcal{Q}_0 = \mathcal{L}(\bar{w}) + g^T(w-\bar{w}) + \frac{1}{2}(w-\bar{w})^T H(w-\bar{w}) \tag{1}$$

minimize $\mathcal{Q}_0(w)$ subject to a cardinality constraint

$$\min_w \mathcal{Q}_0(w) \qquad s.t. \quad ||w||_0 \leq k \tag{2}$$

Related Work

- OBD (Optimal Brain Damage) framework : it is usually assumed that $\bar{w}$ is a local optimum of $\mathcal{L}$ and $g = 0$, searches for a single weight i to prune with minimal increase of the loss function,
- OBS (Optimal Brain Surgeon): use empirical Fisher information matrix to approximate Hessian on a small subset of the training data $(n << N)$ $\nabla^2\mathcal{L}(\bar{w}) \approx H = \frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\bar{w})\nabla l_i(\bar{w})^T$

  OBD and OBS do not to consider the possible interactions that can arise when pruning multiple weights.
- approximate the gradient by the stochastic gradient, using the same samples for estimating the Hessian $g = \frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\bar{w})$
- *one-shot pruning* methods can be followed by a few fine-tuning and re-training steps to recover some of the accuracy lost when pruning (*gradual pruning*).

Outline

l_0-regression formulation

The formulation is based on a critical observation that the Hessian approximation in has a low-rank structure:

$$H = \frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\bar{w})\nabla l_i(\bar{w})^T = \frac{1}{n}A^T A \in \mathbb{R}^{p\times p} \tag{3}$$

$A = [\nabla l_1(\bar{w}), \dots, \nabla l_n(\bar{w})]^T \in \mathbb{R}^{n\times p}$ has rank at most $n << p$

l_0-regression formulation

The formulation is based on a critical observation that the Hessian approximation in has a low-rank structure:

$$H = \frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\bar{w}) \nabla l_i(\bar{w})^T = \frac{1}{n} A^T A \in \mathbb{R}^{p \times p} \tag{3}$$

$A = [\nabla l_1(\bar{w}), \dots, \nabla l_n(\bar{w})]^T \in \mathbb{R}^{n \times p}$ has rank at most $n << p$

Problem (2) can be equivalently written in the following Hessian-free form ($b = A\bar{w} - e$)

$$\min_w \frac{1}{2} ||b - Aw||^2 \quad s.t. \quad ||w||_0 \leq k \tag{4}$$

To improve solution quality, include a ridge-like regularizer to the objective in (4)

$$\min_w Q(w) = \frac{1}{2} ||b - Aw||^2 + \frac{n\lambda}{2} ||w - \bar{w}||^2 \quad s.t. \quad ||w||_0 \leq k \tag{5}$$

the success of final pruned model depends heavily on the accuracy of the quadratic approximation of the loss function. One way to achieve this is by including a squared l_2 penalty, also known as the ridge, on the difference $w - \bar{w}$.

Algorithm design

The optimization framework relies on the IHT (Iterative Hard Thresholding) algorithm (slow for large parameters). They propose a new line search scheme and use an active set strategy to update the weights on the nonzero weights upon support stabilization.

Algorithm design

STRUCTURE-AWARE IHT UPDATE

- for any vector x, $\mathcal{I}_k(x)$ denotes the indices of k components of x that have the largest absolute value.
- hard thresholding operator $P_k(x) = \begin{cases} x_i, \text{ if } i \in \mathcal{I}_k(x) \\ 0, \text{ otherwise} \end{cases}$ for each i-th coordinate of $P_k(x)$.
- IHT applied to problem (5) leads to the following update:

$$\begin{aligned} w^{t+1} &= HT(w^k, k, \tau^s) = P_k(w^t - \tau^s \nabla Q(w^t)) \\ &= P_k\Big(w^t - \tau^s\big(A^T(Ab - w^t) + n\lambda(w^t - \bar{w})\big)\Big) \end{aligned} \tag{6}$$

  where τ^s is a suitable stepsize. The computation of $HT(w^k, k, \tau^s)$ is in $O(np)$ $(n << p)$.

Active set strategy restricts the IHT updates to an *active set* (a relatively small subset of variables) and occasionally augmenting the active set with variables that violate certain optimality conditions.

Algorithm design

DETERMINING A GOOD STEPSIZE

Appropriate stepsize τ^s is crucial for fast convergence of the IHT algorithm. A common choice is to use a constant stepsize of $\tau^s = \frac{1}{L}$, where L is the Lipschitz constant of the gradient of the objective function.

Backtracking line search starts with a relatively large estimate of the stepsize and iteratively shrinking the step size until a sufficient decrease of the objective function is observed. However multiple evaluations of the objective function can be computationally expensive.

Algorithm design

DETERMINING A GOOD STEPSIZE

Appropriate stepsize τ^s is crucial for fast convergence of the IHT algorithm. A common choice is to use a constant stepsize of $\tau^s = \frac{1}{L}$, where L is the Lipschitz constant of the gradient of the objective function.

Backtracking line search starts with a relatively large estimate of the stepsize and iteratively shrinking the step size until a sufficient decrease of the objective function is observed. However multiple evaluations of the objective function can be computationally expensive.

novel line search method improves the convergence speed of IHT, finds the stepsize that leads to the maximum decrease in the objective.

$$\min_{\tau^s \geq 0} \quad g(\tau^s) = Q\Big(P_k\big(w^t - \tau^s \nabla Q(w^t)\big)\Big) \tag{7}$$

$g(\tau^s)$ is a piecewise quadratic function

Algorithm design

DETERMINING A GOOD STEPSIZE

Appropriate stepsize τ^s is crucial for fast convergence of the IHT algorithm. A common choice is to use a constant stepsize of $\tau^s = \frac{1}{L}$, where L is the Lipschitz constant of the gradient of the objective function.

Backtracking line search starts with a relatively large estimate of the stepsize and iteratively shrinking the step size until a sufficient decrease of the objective function is observed. However multiple evaluations of the objective function can be computationally expensive.

novel line search method improves the convergence speed of IHT, finds the stepsize that leads to the maximum decrease in the objective.

$$\min_{\tau^s \geq 0} \quad g(\tau^s) = Q\Big(P_k\big(w^t - \tau^s \nabla Q(w^t)\big)\Big) \tag{7}$$

$g(\tau^s)$ is a piecewise quadratic function

The single-stage algorithm CHITA takes as input a low-rank matrix A, the initial weight $\bar{w}$ and the l_0-constraint k; and returns a pruned weight w that serves as a good solution to (5).

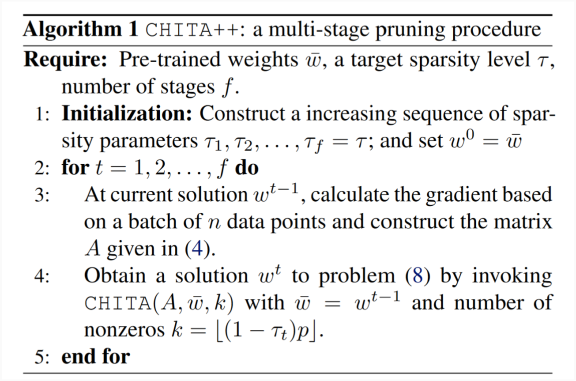
Multi-stage CHITA++

The final performance/accuracy of the pruned network depends heavily on the quality of the local quadratic approximation (especially for high levels of sparsity).

Multi-stage CHITA++

The final performance/accuracy of the pruned network depends heavily on the quality of the local quadratic approximation (especially for high levels of sparsity).

CHITA++ gradually increases the sparsity constraint and takes a small step towards higher sparsity in each stage to ensure the validity of the local quadratic approximation. (unlike other gradual pruning approach that also includes fine-tuning steps in which SGD is applied to further optimize the parameters for better results) multi-stage method is a one-shot pruning method and only requires constructing and solving Problem (5)



```
Algorithm 1 CHITA++: a multi-stage pruning procedure
Require: Pre-trained weights $\bar{w}$, a target sparsity level $\tau$,
    number of stages $f$.
1: Initialization: Construct a increasing sequence of sparsity parameters $\tau_1, \tau_2, \ldots, \tau_f = \tau$; and set $w^0 = \bar{w}$
2: for $t = 1, 2, \ldots, f$ do
3:     At current solution $w^{t-1}$, calculate the gradient based on a batch of $n$ data points and construct the matrix $A$ given in (4).
4:     Obtain a solution $w^t$ to problem (8) by invoking CHITA$(A, \bar{w}, k)$ with $\bar{w} = w^{t-1}$ and number of nonzeros $k = \lfloor (1 - \tau_t) p \rfloor$.
5: end for
```

Outline

1. Problem Setup and Related Work

2. CHITA (Combinatorial Hessian-free Iterative Thresholding Algorithm)

3. Experimental Results
    - Compare with one shot pruning approaches
    - Performance on gradual pruning

RUNTIME COMPARISON

- MP (Magnitude Pruning)
- WF (WoodFisher)
- CBS (Combinatorial Brain Surgeon)
- M-FAC (Matrix-Free Approximate Curvature)

RUNTIME COMPARISON

Using more samples for Hessian and gradient approximation results in better accuracy.

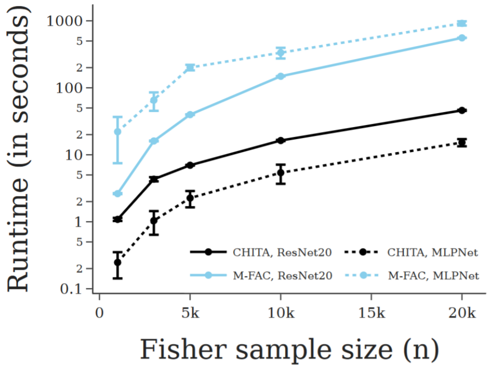
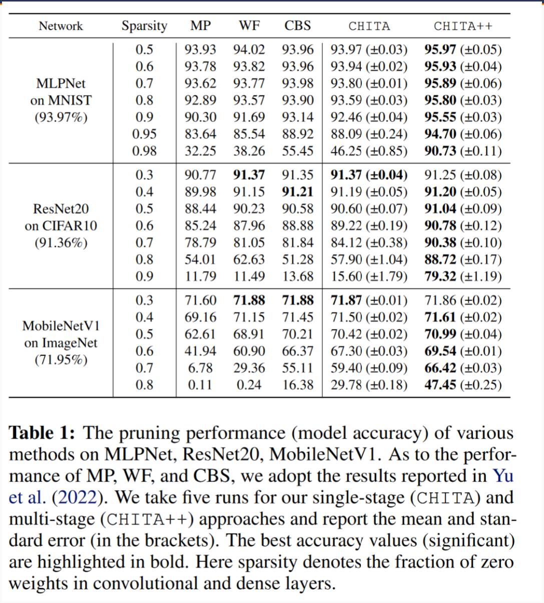


Figure 1: Runtime comparison between our single-stage approaches and M-FAC (the fastest among the competitive methods) while pruning MLPNet and ResNet20 to 90% sparsity level (90% of the entries are zero). Note that Woodfisher and CBS are at least 1000 times slower than M-FAC. The error bar represents the standard error over four runs. CHITA here uses IHT to find a support and performs a back-solve on the found support.

ACCURACY OF THE PRUNED MODELS

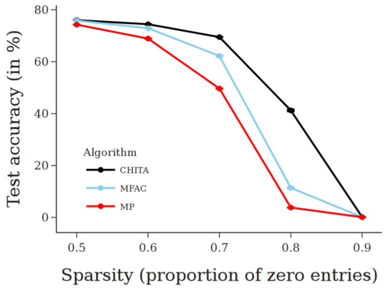


Network	Sparsity	MP	WF	CBS	CHITA	CHITA++
MLPNet on MNIST (93.97%)	0.5	93.93	94.02	93.96	93.97 (±0.03)	**95.97** (±0.05)
	0.6	93.78	93.82	93.96	93.94 (±0.02)	**95.93** (±0.04)
	0.7	93.62	93.77	93.98	93.80 (±0.01)	**95.89** (±0.06)
	0.8	92.89	93.57	93.90	93.59 (±0.03)	**95.80** (±0.03)
	0.9	90.30	91.69	93.14	92.46 (±0.04)	**95.55** (±0.03)
	0.95	83.64	85.54	88.92	88.09 (±0.24)	**94.70** (±0.06)
	0.98	32.25	38.26	55.45	46.25 (±0.85)	**90.73** (±0.11)
ResNet20 on CIFAR10 (91.36%)	0.3	90.77	**91.37**	91.35	**91.37 (±0.04)**	91.25 (±0.08)
	0.4	89.98	91.15	**91.21**	91.19 (±0.05)	**91.20** (±0.05)
	0.5	88.44	90.23	90.58	90.60 (±0.07)	**91.04** (±0.09)
	0.6	85.24	87.96	88.88	89.22 (±0.19)	**90.78** (±0.12)
	0.7	78.79	81.05	81.84	84.12 (±0.38)	**90.38** (±0.10)
	0.8	54.01	62.63	51.28	57.90 (±1.04)	**88.72** (±0.17)
	0.9	11.79	11.49	13.68	15.60 (±1.79)	**79.32** (±1.19)
MobileNetV1 on ImageNet (71.95%)	0.3	71.60	**71.88**	**71.88**	**71.87** (±0.01)	71.86 (±0.02)
	0.4	69.16	71.15	71.45	71.50 (±0.02)	**71.61** (±0.02)
	0.5	62.61	68.91	70.21	70.42 (±0.02)	**70.99** (±0.04)
	0.6	41.94	60.90	66.37	67.30 (±0.03)	**69.54** (±0.01)
	0.7	6.78	29.36	55.11	59.40 (±0.09)	**66.42** (±0.03)
	0.8	0.11	0.24	16.38	29.78 (±0.18)	**47.45** (±0.25)

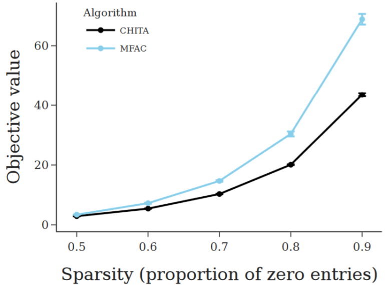
Table 1: The pruning performance (model accuracy) of various methods on MLPNet, ResNet20, MobileNetV1. As to the performance of MP, WF, and CBS, we adopt the results reported in Yu et al. (2022). We take five runs for our single-stage (CHITA) and multi-stage (CHITA++) approaches and report the mean and standard error (in the brackets). The best accuracy values (significant) are highlighted in bold. Here sparsity denotes the fraction of zero weights in convolutional and dense layers.

Our single-stage method achieves comparable results to other state-of-the-art approaches with much less time consumption.

One-shot pruning on ResNet50 (large network)



(a) Test accuracy for one-shot pruning on ResNet50.



(b) The objective value in (8) for pruning ResNet50.

CHITA achieves a lower objective value, and in this case, it also results in a better test accuracy

Sparsity schedule in multi-stage procedure

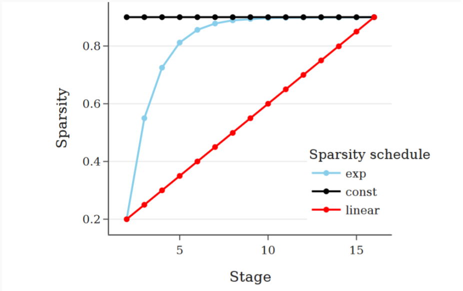
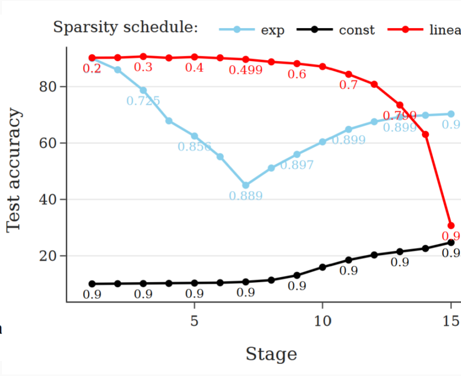


Figure 3: Three different sparsity schedules: exponential, lin
and constant schedules.



Sparsity schedule in multi-stage procedure

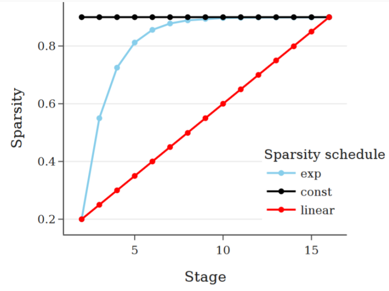
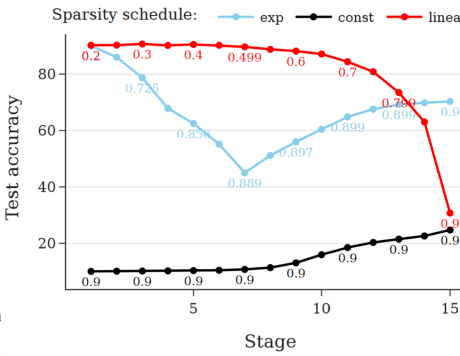


Figure 3: Three different sparsity schedules: exponential, lin
and constant schedules.

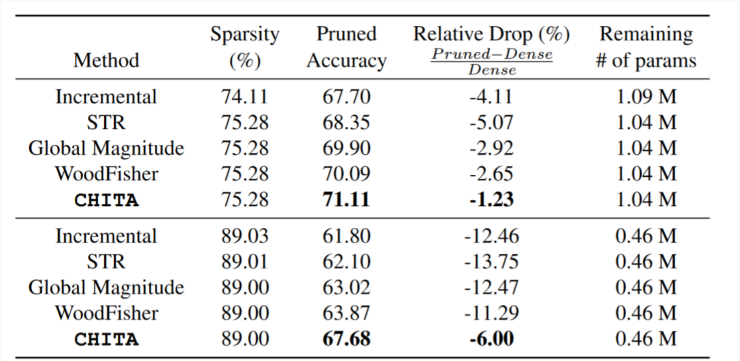


linear mesh outperforms the exponential mesh in the first few iterations, but its performance drops dramatically in the last two iterations. The reason is that in high sparsity levels, even a slight increase in the sparsity rate leads to a large drop in accuracy.

Taking small “stepsizes” in high sparsity levels allows the exponential mesh to fine-tune the weights in the last several stages and achieve good performance.

Performance on gradual pruning

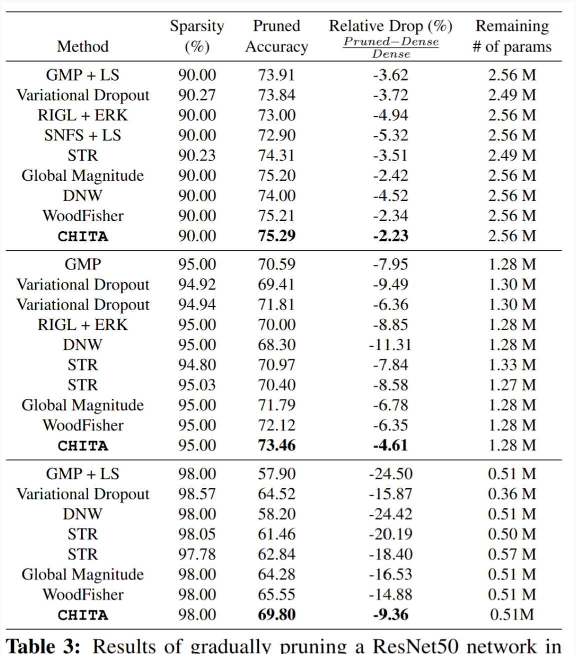
Alternate between pruning steps where a sparse weight is computed and fine-tuning steps on the current support via Stochastic Gradient Descent (SGD).



Method	Sparsity (%)	Pruned Accuracy	Relative Drop (%) $\frac{Pruned-Dense}{Dense}$	Remaining # of params
Incremental	74.11	67.70	-4.11	1.09 M
STR	75.28	68.35	-5.07	1.04 M
Global Magnitude	75.28	69.90	-2.92	1.04 M
WoodFisher	75.28	70.09	-2.65	1.04 M
CHITA	75.28	**71.11**	**-1.23**	1.04 M
Incremental	89.03	61.80	-12.46	0.46 M
STR	89.01	62.10	-13.75	0.46 M
Global Magnitude	89.00	63.02	-12.47	0.46 M
WoodFisher	89.00	63.87	-11.29	0.46 M
CHITA	89.00	**67.68**	**-6.00**	0.46 M

Table 2: Results of gradually pruning MobilenetV1 in 75% and 89% sparsity regimes, comparing CHITA to other baselines (Dense accuracy: 72.00%). We also include the relative drop in accuracy to account for different methods starting from different dense weights. CHITA numbers are averaged across two runs. Numbers for other baselines are taken from Singh & Alistarh (2020).

Performance on gradual pruning



Method	Sparsity (%)	Pruned Accuracy	Relative Drop (%) $\frac{Pruned-Dense}{Dense}$	Remaining # of params
GMP + LS	90.00	73.91	-3.62	2.56 M
Variational Dropout	90.27	73.84	-3.72	2.49 M
RIGL + ERK	90.00	73.00	-4.94	2.56 M
SNFS + LS	90.00	72.90	-5.32	2.56 M
STR	90.23	74.31	-3.51	2.49 M
Global Magnitude	90.00	75.20	-2.42	2.56 M
DNW	90.00	74.00	-4.52	2.56 M
WoodFisher	90.00	75.21	-2.34	2.56 M
CHITA	90.00	**75.29**	**-2.23**	2.56 M
GMP	95.00	70.59	-7.95	1.28 M
Variational Dropout	94.92	69.41	-9.49	1.30 M
Variational Dropout	94.94	71.81	-6.36	1.30 M
RIGL + ERK	95.00	70.00	-8.85	1.28 M
DNW	95.00	68.30	-11.31	1.28 M
STR	94.80	70.97	-7.84	1.33 M
STR	95.03	70.40	-8.58	1.27 M
Global Magnitude	95.00	71.79	-6.78	1.28 M
WoodFisher	95.00	72.12	-6.35	1.28 M
CHITA	95.00	**73.46**	**-4.61**	1.28 M
GMP + LS	98.00	57.90	-24.50	0.51 M
Variational Dropout	98.57	64.52	-15.87	0.36 M
DNW	98.00	58.20	-24.42	0.51 M
STR	98.05	61.46	-20.19	0.50 M
STR	97.78	62.84	-18.40	0.57 M
Global Magnitude	98.00	64.28	-16.53	0.51 M
WoodFisher	98.00	65.55	-14.88	0.51 M
CHITA	98.00	**69.80**	**-9.36**	0.51M

Table 3: Results of gradually pruning a ResNet50 network in

1. Benbaki, R. *et al.* **Fast as CHITA: Neural Network Pruning with Combinatorial Optimization.** *arXiv preprint arXiv:2302.14623* (2023).
2. Yu, X., Serra, T., Ramalingam, S. & Zhe, S. ***The combinatorial brain surgeon: pruning weights that cancel one another in neural networks.*** in *International Conference on Machine Learning* (2022), 25668–25683.