Review of Sparsity

이종진

Seoul National University

ga0408@snu.ac.kr

July 19, 2021

Table of Contents

Necessities

- Neural Network success on Computer vision / Speech recognition / Language processing
- These has accompanied by a significant increase in the computation and parameter storage costs
- Overparameterization has been shown to benefit both the optimization and generalization of neural networks.

Necessities

- resource-constrained environments
  - Mobile devices, wearable devices, IoT
  - Reducing the storage footprint
  - Reducing the computation cost of inference
  - Reducing the energy requirements of inference(battery constrained devises)
- Compression methods indicates the existence of compact network parameter configurations.
  - Better generalization bound
  - Alternative training methods might exist to discover and train compact networks directly.
- (Fewer training examples required)

Fomulate problem settings

- Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ and a desired sparsity level κ

$$\min_{\theta} L(\theta; \mathcal{D}) = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f\left(\mathrm{x}_i; \theta\right), \mathrm{y}_i\right),$$

$$\text{s.t.} \quad \theta \in \mathbb{R}^p, \quad \|\theta\|_0 \leq \kappa.$$

- With auxiliary indicator variables $m = \{0, 1\}^p$ (connectivity/mask/gate)

$$\min_{\mathrm{m}, \theta} L(\mathrm{m} \odot \theta; \mathcal{D}) = \min_{\mathrm{m}, \theta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f\left(\mathrm{x}_i; \mathrm{m} \odot \theta\right), \mathrm{y}_i\right),$$

$$\text{s.t.} \quad \mathrm{w} \in \mathbb{R}^p, \quad \mathrm{m} \in \{0, 1\}^p, \quad \|\mathrm{m}\|_0 \leq \kappa,$$

Measures/Evaluation metrics

- Sparsity, Nonzero rate
- FLOPS
  - # of parameters: FC $>$ Conv-layers
  - FLOPS drop by pruning parameters: FC $<$ Conv-layers
- A trade-off between model quality and efficiency.
  - A family of models corresponding to different points on the efficiency-quality curve
- Most paper report changes of accuracy in multiple efficiency points

Methods

- 1. Compression
- 2. Sparsity constrain/regularization
- 3. Sparse Bayesian Learning
- 4. Dynamic sparse training
- 5. Others (Qunatization / Binarization) / (Weight sharing) / (Knowledge distillation) / (Specialized structure)

1. Compression

- It consists of three steps:
  1. Training
  2. Pruning
  3. Fine-tuning
- Heuristically designed pruning schedules depending on the dataset and architecture dependence.
- (LeCun et al., 1990) / (HASSIBI, 1993) / (Han et al., 2015a) / (Li et al., 2016)/ (Frankle and Carbin, 2018) / (Liu et al., 2018) / (Lee et al., 2018) / (Wang et al., 2019) / (Wang et al., 2020)

1. Compression

- Training
  - Pre-train
- Pruning
  - Importance / saliency measures
  - Pruning units (unstructure / structure)
  - Single shot (one shot) / iterative
  - Locally / Globally
  - Retain / Reinitialize / Reinitial
- Fine-tuning

Papers; Magnitude

- SongHan (Han et al., 2015a)[1] / Lottery (Frankle and Carbin, 2018)[2] / Rethinking (Liu et al., 2018) [3] / (Efficient convs (Li et al., 2016)) [4]
- The importance of each weight is measured by its magnitude ($|\theta|$).
- The main difference is
  - Retain (Han et al., 2015a)
  - Rewind (Frankle and Carbin, 2018)
  - Reinitialize (Liu et al., 2018)
- Pre-train, unstructured, locally, iteratively.

[1] Learning Both weights and Connections for Efficient Neural Networks.
[2] The Lottery Tickey Hypothesis Finding Sparse, Trainable Neural Networks.
[3] Rethinking the value of Network Pruning.
[4] Pruning Filters for Efficient Convnets

Papers; Hessian based / Loss preserving

- Loss preserving

$$\Delta\mathcal{L} = \underbrace{\frac{\partial\mathcal{L}}{\partial\theta}^{\top}\Delta\theta}_{\approx 0} + \frac{1}{2}\Delta\theta^{\top}\mathrm{H}\Delta\theta + \mathcal{O}\left(\|\Delta\theta\|^{3}\right)$$

- $\Delta\theta = -\theta_q$
- OBD (LeCun et al., 1990)[5] /OBS (HASSIBI, 1993)[6] /EigenDamge (Wang et al., 2019) [7]

[5] Optimal Brain Damage

[6] Second Order Derivatives for Network Pruning, Optimal Brain Surgeon.

[7] EigenDamage, Structured Prunning in the Kronecker-Factored Eigenbasis

Papers; Hessian based / Loss preserving

- In OBD

$$\Delta\theta_q = -\theta_q^* \text{ and } \Delta\mathcal{L}_{\text{OBD}} = \frac{1}{2}\left(\theta_q^*\right)^2 \mathrm{H}_{qq}$$

  In OBS, the importance of each weight is calculated by solving the following constrained optimization problem:

$$\min_q \left\{ \min_{\Delta\theta} \frac{1}{2} \Delta\theta^\top \mathrm{H} \Delta\theta \quad \text{s.t. } \mathrm{e}_q^\top \Delta\theta + \theta_q^* = 0 \right\}$$

- OBD: $\Delta\mathcal{L}_{\mathrm{OBD}} = \frac{1}{2}\left(\theta_q^*\right)^2 \mathrm{H}_{qq}$ / OBS: $\Delta\mathcal{L}_{\mathrm{OBS}} = \frac{1}{2}\frac{\left(\theta_q^*\right)^2}{\left[\mathrm{H}^{-1}\right]_{qq}}$
- OBD has diagonal assumption on Hessian matrix.
- OBS does not require fine-tunnig after pruning.
- Pre-train, unstructured, globally, iteratively/one-shot).

Papers; Hessian based / Loss preserving

- EigenDamage (Wang et al., 2019)
- They propose Kron-OBD, Kron-OBS and EigenDamage
- Kron-OBD, Kron-OBS extend OBS, OBD to filterwise pruning
- Take into account the correlation of the weights within the same filter

$$\Delta \mathcal{L}_j = \frac{1}{2} \mathcal{F}_j^{l*\top} \mathrm{H}^l(j) \mathcal{F}_j^{l*}$$

  where $F_j^{l*} \in \mathbb{R}^{c_{\mathbf{in}}\ k^2}$ and $\mathrm{H}^l(j) \in \mathbb{R}^{c_{\mathbf{in}}\ k^2 \times c_{\mathbf{in}}\ k^2}$

- Kron-OBS considers the correlation between filters.

Papers; Hessian based / Loss preserving

- It cannot be applied for large convolutional layers, they adopt K-FAC approximation.
- $\mathsf{F} = \mathsf{S} \otimes \mathsf{A}$ where $\mathsf{A} = \mathbb{E}\left[\mathsf{a}\mathsf{a}^\top\right]$ and $\mathsf{S} = \mathbb{E}\left[\{\nabla_{\mathsf{s}}\mathcal{L}\}\{\nabla_{\mathsf{s}}\mathcal{L}\}^\top\right]$,
- Kron-OBD

$$\Delta F_j^{l*} = -F_j^{l*} \text{ and } \Delta\mathcal{L}_j = \frac{1}{2}\mathsf{S}_{jj} F_j^{l*\top} \mathsf{A} F_j^{l*}$$

- Kron-OBS

$$\Delta F_j^{l*} = -\frac{\mathsf{S}^{-1}\mathsf{e}_j \otimes F_j^{l*}}{[\mathsf{S}^{-1}]_{ii}} \text{ and } \Delta\mathcal{L}_j = \frac{1}{2}\frac{F_j^{l*\top}\mathsf{A}F_j^{l*}}{[\mathsf{S}^{-1}]_{jj}}$$

- Pre-train, structured, locally, iteratively.

Papers; Hessian based / Loss preserving

- EigenDamage, Decorrelate the weights before pruning.
- Hessian matrix is closer to diagonal in the KFE, we can apply OBD with less cost to prediction accuracy.
- Low-rank decomposition. ((Lebedev et al., 2015)[8]/ (Jaderberg et al., 2014)[9]/ (Denton et al., 2014)[10])

[8]Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition
[9]Speeding up Convolutional Neural Networks with Low Rank Expansions
[10]Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation

Papers; train dynamics preserve, pre-train

- SNIP (Lee et al., 2018)[11] / Grasp (Wang et al., 2020) [12] / Signal Propagation (Lee et al., 2019) [13]
- These methods does not need pre-training
- Prune parameters reserving train dynamics.
- Need not pre-train, unstructured, globally, one-shot

[11] SNIP, Single-Shot Network Pruning Based on Connection Sensitivity

[12] Picking Winning Tickets Before Training by Preserving Gradient Flow

[13] A Signal Propagation Perspective for Pruning Neural Networks at Initialization.

Papers; train dynamics preserve, pre-train

- For data dependency, they use gradient information.
- SNIP: $S\left(\theta_q\right) = \lim_{\epsilon \to 0} \left| \frac{\mathcal{L}(\theta_{\mathbf{0}}) - \mathcal{L}\left(\theta_{\mathbf{0}} + \epsilon \delta_q\right)}{\epsilon} \right| = \left| \theta_q \frac{\partial \mathcal{L}}{\partial \theta_q} \right|$
- Grasp:

$$\mathrm{S}(\boldsymbol{\delta}) = \Delta\mathcal{L}\left(\boldsymbol{\theta}_0 + \boldsymbol{\delta}\right) - \underbrace{\Delta\mathcal{L}\left(\boldsymbol{\theta}_0\right)}_{\text{Const}} = 2\delta^\top \nabla^2 \mathcal{L}\left(\boldsymbol{\theta}_0\right) \nabla \mathcal{L}\left(\boldsymbol{\theta}_0\right) + \mathcal{O}\left(\|\delta\|_2^2\right)$$

$$= 2\delta^\top \mathrm{Hg} + \mathcal{O}\left(\|\delta\|_2^2\right),$$

Papers; Structured pruning

- Efficient convs (Li et al., 2016)
- Importance of filters is defined the sum of weights in filters.
- For each filter $\mathcal{F}_{i,j}$, calculate the sum of its absolute kernel weights $s_j = \sum_{l=1}^{n_i} \sum |\mathcal{K}_l|$ Sort the filters by s_i
- Pre-train, structured, locally, iteratively

2. Regularization

- Loss + model complexity.
- L_0, L_1(LASSO), $L_{2,1}$(Group LASSO), L_2 (Weight-decay)
- Regularization on weights / on structure related factor.
- (Tartaglione et al., 2018)[14] / (Liu et al., 2017) / (Huang and Wang, 2018)[15] / (Zhu et al.)[16]/(Louizos et al., 2018) [17]

[14] Learning Sparse Neural Networks via Sensitivity-Driven Regularization
[15] Data-Driven Sparse Structure Selection for Deep Neural Networks.
[16] Improving Deep Neural Network Sparsity through Decorrelation Regularization.
[17] Learning Sparse Neural Networks Through L0 Regularzation

Papers; Group LASSO

- (Liu et al., 2017)
- Our approach impose $L1$ regularization on the scaling factors in batch normalization layers.
- Training objective

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i;\theta),y_i)+\lambda\sum_{\gamma}g(\gamma)$$

- BN

$$\hat{z}=\frac{z_{in}-\mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}};z_{\text{out}}\ =\gamma\hat{z}+\beta$$

- Global Threshold across all layers.
- Prune those channels with small factors and fine-tune the pruned network.

Papers; Group LASSO

- (Huang and Wang, 2018)
- Introduce scaling factors which sacle the outputs of specific structure(e.g. neurons, groups or blocks)
- Add sparsity regularizations on scaling factors.
- Try to enforce the output of the group zero.
- Better results without fine-tuning and multi-stage optimization
- Training objective

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i;\theta), y_i) + \lambda\|\theta\|_2^2 + R_s(\gamma)$$

Papers; Group LASSO

- (Zhu et al.)
- Minimizing the filter corrleation during training.
- Training objective function

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i;\theta), y_i) + \lambda\|\theta\|_2^2 + \eta \cdot R_S + \gamma \cdot \sum_{l=1}^{L} R_C^l$$

- R_s: group LASSO regularization for filter groups
- $R_C^l = \frac{1}{\sum_k M_k^l}\left\|\hat{\mathrm{C}}^l - \mathrm{I}\right\|_2^2$, where $\sum M_K^l$ denotes number of unmasked filter and $\hat{\mathrm{C}}^l$ is the correlation matrix of the unmasked filters in layer l

Papers; L_2

- Sensitivity-Driven (Tartaglione et al., 2018)
- Gradually lowers the absolute value of parameters with low sensitivity.
- Eventually set to zero by simple thresholding.
- Sensitivity

$$S\left(\boldsymbol{y}, w_{n,i}\right) = \sum_{k=1}^{C} \alpha_k \left| \frac{\partial y_k}{\partial w_{n,i}} \right|$$

- Insensitivity

$$\bar{S}\left(\mathrm{y}, w_{n,i}\right) = 1 - S\left(\mathrm{y}, w_{n,i}\right)$$

- A Bounded insensitivity

$$\bar{S}_b\left(\mathrm{y}, w_{n,i}\right) = \max\left[0, \bar{S}\left(\mathrm{y}, w_{n,i}\right)\right]$$

Papers; L_2

- Update rule

$$w_{n,i}^{t} := w_{n,i}^{t-1} - \eta \frac{\partial L}{\partial w_{n,i}^{t-1}} - \lambda w_{n,i}^{t-1} \bar{S}_b \left(\mathrm{y}, w_{n,i}^{t-1} \right)$$

- Regularization

$$R_{n,i}\left(w_{n,i}\right) = \frac{w_{n,i}^2}{2} \bar{S}\left(\mathrm{y}, w_{n,i}\right)$$

- If $\alpha_k = \frac{1}{C}$
- Specific (data-driven)

$$S^{spec}\left(\mathrm{y}, \mathrm{y}^*, w_{n,i}\right) = \sum_{k=1}^{C} y_k^* \left| \frac{\partial y_k}{\partial w_{n,i}} \right|$$

- Pruning with global threshold T.

Papers; L_0

- L_0 (Louizos et al., 2018)
- Propose a general framework for surrogated L_0 regularized objectives
- It is realized by smoothing the expected L_0 regularized objectives with continuous distributions.

Papers; L_0

- With auxiliary indicator variables $m = \{0, 1\}^p$

$$\min_{\mathrm{m},\theta} L(\mathrm{m} \odot \theta; \mathcal{D}) = \min_{\mathrm{m},\theta} \left(\frac{1}{n} \sum_{i=1}^{n} \ell \left(f \left(\mathrm{x}_i; \mathrm{m} \odot \theta \right), \mathrm{y}_i \right) + \lambda \|\mathrm{m}\|_0 \right),$$

$$\text{s.t.} \quad \mathrm{w} \in \mathbb{R}^p, \quad \mathrm{m} \in \{0, 1\}^p$$

- By letting $q(m_j|\pi_j) = \mathrm{Ber}(\pi_j)$

$$\min_{\pi,\theta} \left(\mathbb{E}_{q(m|\pi)} \left[\frac{1}{n} \sum_{i=1}^{n} \ell \left(f \left(\mathrm{x}_i; \mathrm{m} \odot \theta \right), y_i \right) \right] + \lambda \sum_{j=1}^{p} \pi_j \right)$$

Papers; L_0

- A hard-sigmoid rectification of s

$$\mathrm{s} \sim q(s \mid \phi)$$

$$\mathrm{m} = \min(1, \max(0, \mathrm{s}))$$

- Then with CDF Q,

$$\min_{\phi,\tilde{\theta}} \left(\mathbb{E}_{q(s|\phi)} \left[\frac{1}{n} \sum_{i=1}^{n} \ell \left(f \left(\mathrm{x}_i; \mathrm{g}(\mathrm{s}) \odot \tilde{\theta} \right), \mathrm{y}_i \right) \right] + \lambda \sum_{j=1}^{p} (1 - Q(\mathrm{s}_j \geq 0 | \phi_j)) \right)$$

Papers; L_0

- With the reparametrization trick,

$$\min_{\phi,\tilde{\theta}} \left(\mathbb{E}_{p(\epsilon)} \left[\frac{1}{n} \sum_{i=1}^{n} \ell \left(f \left(\mathrm{x}_i; \mathrm{g}(\mathrm{h}(\phi, \epsilon)) \odot \tilde{\theta} \right), \mathrm{y}_i \right) \right] + \lambda \sum_{j=1}^{p} (1 - Q(\mathrm{s}_j \geq 0 | \phi_j)) \right)$$

- Monte Carlo approximation,

$$\min_{\phi,\tilde{\theta}} \left(\frac{1}{L} \sum_{l=1}^{L} \left(\frac{1}{n} \sum_{i=1}^{n} \ell \left(f \left(\mathrm{x}_i; \mathrm{m}^{(\mathrm{l})} \odot \tilde{\theta} \right), \mathrm{y}_i \right) \right) + \lambda \sum_{j=1}^{p} (1 - Q(\mathrm{s}_j \geq 0 | \phi_j)) \right),$$

$$= \mathcal{L}_E(\tilde{\theta}, \phi) + \lambda \mathcal{L}_C(\phi), \quad \text{where } \boldsymbol{m}^{(l)} = g(h(\phi, \epsilon^{(l)})) \text{ and } \epsilon^{(l)} \sim p(\epsilon)$$

3. Sparse Bayesian learning

- Sparse variational dropout (Molchanov et al., 2017)[18]
- The original version of dropout, $\xi_{mi} \sim \text{Bernoulli}(1-p)$
- Gaussian Dropout with multiplicative continuous noise

  $\xi_{mi} \sim \mathcal{N}(1, \alpha = \frac{p}{1-p})$
- It is equivalent to sampling θ_{ij} from $q(\theta_{ij} \mid w_{ij}, \alpha) = \mathcal{N}(\theta_{ij} \mid w_{ij}, \alpha_{ij}^2)$.
- Use $q(\theta_{ij} \mid w_{ij}, \alpha)$ as an approximate posterior distribution
- The parameters (w, α) are tuned via stochastic variational inference.

[18] Variational Dropout Sparsifies Deep Neural Networks.

3. Sparse Bayesian learning

- Original variational dropout paper only consider $a \leq 1$.
- Extend it to the case, $\alpha > 1$
- $\alpha \to \infty$ corresponds to $p \to 1$ which means it could be always dropped from the model.
- Extended variational dropout leads to extremely sparse solutions both in fully-connected and convolutional layers.

4. Dynamic sparse training

- Dynamically explore structures during training within sparsity constraint.
- Taking inspiration from the biological neural networks which have a sparse topology.
- Naturally, it does not require pre-training
- Pruning methods dealt with previously could be called static sparse training.
- (Mocanu et al., 2018)[19] / (Bellec et al., 2018)[20] / (Mostafa and Wang, 2019)[21]

[19] Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science

[20] Deep Rewiring, Training Very Sparse Deep Networks

[21] Parameter Efficient Training of Deep Convolutional Neural Networks by Dynamic Sparse Reparameterization.

Papers; SET

- SET(Sparse Evolutionary Training) (Mocanu et al., 2018)
- With kth layer has n^k neurons, matrix $\boldsymbol{W}^k \in \boldsymbol{R}^{n^k \times n^{k-1}}$
- ϵ is a parameter of SET controlling the sparsity level.
- The probability of a connection between the neuron h_i^k and h_j^{k-1} is

$$p(m_{ij}^k = 1) = \frac{\epsilon(n^k + n^{k-1})}{n^k n^{k-1}}$$

- $\epsilon(n^k + n^{k-1})$ numbers of weights is expected to be not zero in each layers.

Papers; SET

1. Initialize a model
2. For each epoch after training:
   2.1 remove a fraction ζ of the smallest positive weights(put $\theta_k = 0$)
   2.2 remove a fraction ζ of the largest positive weights(put $\theta_k = 0$)
   2.3 If it is not last training epoch, then add randomly(uniformly) chosen weights.

Papers; DEEP R

- DEEP R(Bellec et al., 2018)
- Training alogorithm that train direcly a sparse connected neural networks.
- DEEP R is differnt from standard GD in two respects
  - When the absolute value of a weight is moved trough 0, it becomes $\theta_k = 0$, and randomly drawn other connection is tried out by the algorithm.
  - It combines random walk in parameter space.

Papers; DEEP R

1. Initialize a model
2. For each step in updating the model:
   2.1 If $\theta_k > 0$ then, $\theta_k \leftarrow \theta_k - \eta \frac{\partial}{\partial \theta_k} E_{\mathbf{X},\mathbf{Y}^*}(\boldsymbol{\theta}) - \eta\alpha + \sqrt{2\eta T}\nu_k$
   2.2 After updating the parameters, if $\theta_k < 0$, then remove the parameters, put $\theta_k = 0$)
   2.3 Activate the same number of connections which removed during updating the models

Papers; Dynamic Sparse Reparametrization

- Dynamic Sparse Reparametrization (Mostafa and Wang, 2019)
- Differentialble represenation; Reparameterize original network by $\phi \in \Phi$ and $\psi \in \Psi$ through $\theta = g(\phi; \psi)$, where g is differentiable w.r.t. ϕ but not necessarily w.r.t. ψ.

$$y = f(x; g(\phi; \psi)) \triangleq f_\psi(x; \phi)$$

- Sparse reparameterization is a special case where g is linear projection and ϕ is the non-zero entries, ψ their indices
- If ψ is adjusted adaptively during training; dynamic reparametrization.

Papers; Dynamic Sparse Reparametrization

- Differences from SET
  - Adaptive threshold for pruning.
  - Automatically reallocate parameters across layer during training.

1. Initialize a model
2. For each epoch after training:
   2.1 Remove weights by a global theshold H
   2.2 Adjust pruning threshold with the number of pruned weights K and tolerance δ
   2.3 remove a fraction ζ of the largest positive weights(put $\theta_k = 0$)
   2.4 If it is not last training epoch, then add randomly(uniformly) chosen weights.
   2.5 Reallocate parameters with the number of $\frac{l_i}{L}K$ in each layer, l_i is number of non-zero weights in layer i

Others

- Quantization
  - (Han et al., 2015b)[22]
  - Reduce the required
  - With weight-shareing, reduce the number of bits required to represent weight.
- Knowledge distillation
  - (Hinton et al., 2015)[23]
- Specialized structure
  - Hashnet (Chen et al., 2015)[24]

---

[22] Deep Compression, Compressing Deep Neural Network with Pruning, Trained Quantization and huffman Coding

[23] Distilling the knowledge in a neural network

[24] compressing neural networks with the hashing trick

5.others

- (Gale et al., 2019)[25]
  - Comparing the methods, Magnitude based pruning / Variational dropout / L_0 regularization.
  - Transformer with WMT 2014 English-to-German / Resnet50 with ImageNetn
- (Zhou et al., 2019)[26]
  - Investigate the lottery ticket's principles
- (Paganini and Forde, 2020) [27]
  - Comparing magnitude pruning methods in details
  - Importance measurs, locally/globlally, unstructured/sturctured, rewind/reinitialize.

[25] The State of Sparsity in Deep Neural Networks.

[26] Deconstructing Lottery Tickets, Zeros, Signs, and the Supermask

[27] On Iterative Neural Network Pruning, Reinitialization, and The Similarity of Masks.

6. Difficulties

- Review paper, (Blalock et al., 2020) [28]
- Absence of benchmarks; datasets, measures, architectures.
  - Few papers compare to another.
  - No methods that have been shown to outperform all existing "state-of-the-art" methods
  - Papers report a wide variety of metrics and operating points
- Heuristically designed pruning schedules, architecture dependency.
  - Methodologies are so inconsistent between papers.
  - Methods from layer years do not consistently outperform methods from earlier years

[28] What is the State of Neural Network Pruning

Reference I

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

B HASSIBI. Second order derivatives for network pruning: optimal brain surgeon. *Advances in Neural Information Processing Systems*, 5:164–171, 1993.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015a.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Reference II

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2018.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International Conference on Machine Learning*, pages 6566–6575. PMLR, 2019.

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

Reference III

V Lebedev, Y Ganin, M Rakhuba, I Oseledets, and V Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings*, 2015.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

Reference IV

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.

Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini. Learning sparse neural networks via sensitivity-driven regularization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3882–3892, 2018.

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*, pages 2736–2744, 2017.

Reference V

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

Xiaotian Zhu, Wengang Zhou, and Houqiang Li. Improving deep neural network sparsity through decorrelation regularization.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

Reference VI

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. *Nature communications*, 9(1):1–12, 2018.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019.

Reference VII

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*, 2015b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294. PMLR, 2015.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *arXiv preprint arXiv:1905.01067*, 2019.

Reference VIII

Michela Paganini and Jessica Forde. On iterative neural network pruning, reinitialization, and the similarity of masks. *arXiv preprint arXiv:2001.05050*, 2020.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.