DIVERSITY BASED EDGE PRUNING OF NEURAL NETWORKS USING DETERMINANTAL POINT PROCESSES

Rupam Acharyya
Department of Mathematics
University at Buffalo
rupamach@buffalo.edu

Boyu Zhang
Department of Computer Science
University of Rochester
bzhang25@u.rochester.edu

Ankani Chattoraj
Department of Brain & Cognitive Science
University of Rochester
achattor@ur.rochester.edu

Shouman Das
Department of Mathematics
University of Rochester
anumoshsad@gmail.com

Daniel Štefankovič
Department of Computer Science
University of Rochester
stefanko@cs.rochester.edu

ABSTRACT

Deep learning architectures with huge number of parameters are often compressed using *pruning* techniques. Two classes of pruning techniques are *node pruning* and *edge pruning*. A fairly recent work established that *Determinantal Point Process (DPP)* based node pruning empirically outperforms competing node pruning methods. However, one prominent appeal of edge pruning over node pruning is the consistent finding in literature is that sparse neural networks (edge pruned) generalize better than dense neural networks (node pruned). Building on these previous work and drawing motivation from *synaptic diversity* in the brain, we propose a novel *diversity-based edge pruning* technique for neural networks using DPP. We then empirically show that *DPP edge pruning* for neural networks outperforms other competing methods (both edge and node) on real data.

1 INTRODUCTION

The primary goal of pruning a neural network is to reduce the number of parameters without reducing the network's performance significantly. A popular pruning method is node pruning where redundant neurons from the network are removed (Hanson & Pratt (1989;?), others include 'Optimal Brain Damage' LeCun et al. (1990) and 'Optimal Brain surgeon' Hassibi & Stork (1993)). A fairly recent work on novel node pruning methods Mariet & Sra (2016), propose a node pruning technique where: a diverse subset of nodes are preserved in a given layer using DPP Macchi (1975); Kulesza et al. (2012), followed by *reweighting*, to compensate contributions of the pruned neurons in the network. Finally, they show that DPP node pruning with reweighting performs better than random and importance node pruning He et al. (2014) on real datasets. Another way of network pruning is edge pruning where the number of connections between nodes are removed to reduce redundancy (He et al. (2014); Han et al. (2015)). Though, node pruning methods have their own appeal, literature suggests that sparse networks obtained after edge pruning outperform dense networks obtained after node pruning (see section 3.2 of a recent review article Blalock et al. (2020) which investigates 81 recent pruning based papers). Edge pruning methods also draw motivation from the huge line of research in computational neuroscience on normative models of the brain (*Efficient Coding* models Doya et al. (2007)) that focus on the idea of reducing redundant representations in the brain. Interestingly, research advances in neurophysiology over the years has shown the existence of diverse synapses in the brain and established that unused synapses are eliminated by the brain to conserve energy and ensure efficient information transmission Chechik et al. (1998).

In this work we build on the DPP based pruning idea from Mariet & Sra (2016), empirical observations in pruning literature and observations in the brain. We model saliency of a particular weight (synapse) in a network as the amount of *diverse* information captured by it. We then investigate empirical potentials of pruning that focus on synaptic (weight/connection/edge) diversity in feedforward neural networks via DPP ('diversity based synaptic pruning'). Finally, we show that DPP based edge pruning method for feedforward networks which uses synaptic diversity to remove redundant edges outperforms competing methods on real data. We also observe that DPP edge pruning finds a sparse network which performs better than the unpruned dense network.

2 PRELIMINARIES

2.1 DETERMINANTAL POINT PROCESS (DPP):

DPP Macchi (1975) is a probability distribution over power set of a ground set $\mathcal{G}$, here finite. DPP is a special case of negatively associated distributions Joag-Dev et al. (1983) which assigns higher probability mass on diverse subsets. Formally, a DPP with a marginal kernel L ($\in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{G}|}$) is: $\mathbb{P}[\mathbf{Y} = Y] = \frac{\det(L_Y)}{\det(L+I)}$, where $Y \subseteq \mathcal{G}$ and L_Y is the principal submatrix defined by the indices of Y. We use k-DPP to denote the probability distribution over subsets of fixed size k.

2.2 NODE PRUNING WITH DPP (MARIET & SRA (2016)):

Mariet & Sra (2016) uses DPP to propose a novel node pruning method for feed forward neural network. They define information at node i of layer l as $\boldsymbol{a}_i^l (= (a_{i1}^l, \ldots, a_{in}^l))$, where a_{ij}^l is the activity of node i of layer l on j^{th} input. Here $\boldsymbol{a}_i^l = g\left(\boldsymbol{b}_i^l\right)$, where $\boldsymbol{b}_i^l = \sum_{j=1}^{n_{l-1}} w_{ji}^{l-1} \boldsymbol{a}_j^{l-1}$ is the information at node i of layer l before activation. A layer is pruned by choosing a subset of hidden nodes using a DPP kernel: $\boldsymbol{L}$ ($= \boldsymbol{L'} + \epsilon \boldsymbol{I}$), where, $\boldsymbol{L'}_{st} = \exp(-\beta \left\|\boldsymbol{a}_s^l - \boldsymbol{a}_t^l\right\|^2)$ and β is a bandwidth parameter. The matrix $\boldsymbol{L}$ is of dimension $n_l \times n_l$, as total number of nodes in layer l is n_l. By the property of DPP, this procedure will keep a diverse subset of nodes for each layer w.r.t. information obtained from the training data. A *reweighting* technique (see Section 2.2 of Mariet & Sra (2016)) is then applied to outgoing edges of retained nodes to compensate for information lost in that layer due to node removal. Note that DIVNET denotes DPP node pruning with reweighting as in Mariet & Sra (2016).

3 EDGE PRUNING USING DETERMINANTAL POINT PROCESS

3.1 DPP KERNEL FOR EDGE PRUNING

We follow Mariet & Sra (2016) to compute the amount of information captured by an edge about the training data. Recall from Section 2 that for node v_j^l of layer l, the activation is given as $\boldsymbol{a}_j^l = g\left(\sum_{i=1}^{n_{l-1}} w_{ij}^{l-1} \boldsymbol{a}_i^{l-1}\right)$, where w_{ij}^{l-1} is the weight of the edge from i^{th} node of layer $l-1$ to j^{th} node of layer l. Hence the information carried from layer $l-1$ to node v_j^l of layer l through the edge e_{ij}^{l-1} is $w_{ij}^{l-1} \boldsymbol{a}_i^{l-1}$. We choose a subset of incoming edges for node v_j^l such that it preserves diverse information about the input. The DPP kernel $\boldsymbol{L}^j$ for the set of incoming edges of v_j^l is defined as,

$$\boldsymbol{L}^j = \boldsymbol{L'}^{(j)} + \epsilon \boldsymbol{I}, \tag{1}$$

where $\boldsymbol{L'}_{st}^{(j)} = \exp(-\beta \left\|w_{sj}^{l-1} \boldsymbol{a}_s^{l-1} - w_{tj}^{l-1} \boldsymbol{a}_t^{l-1}\right\|^2)$. Here β is a hyperparameter and ϵ denotes small perturbations to the diagonal of the kernel matrix making it strictly positive definite (as in Mariet & Sra (2016)).

3.2 RETRIEVING PRUNED INFORMATION BY REWEIGHTING

We lose information passed from one layer to the next by removing a subset of edges between them after pruning. To maximize the total amount of information preserved, we reweight the surviving

edges. We denote $S_j = \{i_1, \ldots, i_{k_e}\}$ as the set of incoming edges chosen for the node v_j^l using k_e-DPP. Denote $\hat{w}_{ij}^{l-1} = w_{ij}^{l-1} + \delta_{ij}^{l-1}$ to be the new weight of the edges after reweighting. Hence to minimize the lost information after pruning, we minimize:

$$\left\| \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1} \boldsymbol{a}_i^{l-1} - \sum_{i \in S_j} \hat{w}_{ij}^{l-1} \boldsymbol{a}_i^{l-1} \right\|_2 = \left\| \sum_{i \in \bar{S}_j} w_{ij}^{l-1} \boldsymbol{a}_i^{l-1} - \sum_{i \in S_j} \delta_{ij}^{l-1} \boldsymbol{a}_i^{l-1} \right\|_2 .$$

This can be rewritten as a linear regression problem i.e.,$\min_{\boldsymbol{x}} \left\| \boldsymbol{y} - A_{l-1} \cdot \boldsymbol{x}^j \right\|_2$, where s^{th} column of A_{l-1} is $\boldsymbol{a}_{i_s}^{l-1}$, $\boldsymbol{x}^j = [\delta_{i_1 j}, \cdots, \delta_{i_{k_e} j}]^T$, and $\boldsymbol{y} = \sum_{i \in \bar{S}_j} w_{ij}^{l-1} \boldsymbol{a}_i^{l-1}$. The closed form solution of this equation is, $\boldsymbol{x} = \left(A_{l-1}^T A_{l-1}\right)^{-1} A_{l-1}^T \boldsymbol{y}$. In node pruning, if a node is deleted from layer l, edges from $l-1$ to l and l to $l+1$ are deleted, affecting both incoming and outgoing information from layer l. Therefore, the focus is to minimize loss in outgoing information from layer l to $l+1$ as is ensured by reweighting of Mariet & Sra (2016). In edge pruning, edges between layer $l-1$ to l get deleted which we try to compensate in our new reweighting scheme by minimizing incoming information loss.

While DPP node pruning requires only one kernel per layer Mariet & Sra (2016), DPP edge pruning will have n_l DPP kernels for l^{th} layer. However, DPP kernels require one-time computation and are not recomputed to predict labels for unseen data. Note that while DPP edge pruning with reweighting inherits all benefits of DIVNET discussed in Section 3.3 and 4 of Mariet & Sra (2016), it performs better than the latter on real data (Section 4). Few important properties inherited by DPP edge kernels due to similar definitions to DIVNET are: (1) The DPP edge kernels and the reweighting coefficients can also be learned from subsampled training data which causes a trade-off between performance, space complexity, and time complexity. (2) DPP edge pruning with reweighting can be incorporated into other methods of network compression to increase diversity.

4 Experiment on Real Data

In this section, we compare our proposed DPP edge pruning with reweighting, DIVNET by Mariet & Sra (2016), random edge pruning with reweighting, and importance edge/node pruning with reweighting on the `MNIST` LeCun et al. (2010) and `CIFAR10` Krizhevsky et al. (2009) datasets. We used the exact same network architectures as in Table 1 of Mariet & Sra (2016) for `MNIST` and `CIFAR10`, respectively. Following Mariet & Sra (2016), we performed all pruning methods on the first layer. We compare the number of parameters as $k_e = \frac{k_n(d_{\text{input}}+h_2)}{h_1} - h_2$ where k_e is the number of edges kept for each node in edge pruning, and k_n is the number of nodes kept in the hidden layer for node pruning; d_{input}, h_1, and h_2 represent the dimension of the input, size of the first hidden layer, and size of the second hidden layer, respectively. As in Mariet & Sra (2016), $h_1 = h_2$. We trained our model until the training error reaches predefined thresholds (Table 1 in Mariet & Sra (2016)) and then perform the pruning. All edge pruning methods use our reweighting scheme in 3.2 while all node pruning methods use reweighting from Mariet & Sra (2016).

4.1 Hyperparameters

We report the hyperparameters used for the results on the `MNIST` and `CIFAR10`. We used that exact same experiment setup (network architectures, error thresholds, etc.) as in Mariet & Sra (2016) for fair and consistent comparisons. We used SGD optimizers, a learning rate of 0.001, and a momentum of 0.9 for traning on both datasets. For `MNIST`, the training batch size was 1000. For `CIFAR10`, the training batch size was 128. All pruning methods were performed 10 times, and we report the means and standard deviations in Figure 1A-B (without reweighting) and Figure 1C-D (with reweighting). The node-to-edge ratio for pruning, which keeps the number of parameters in the pruned network the same, is $[397 : 614, 472 : 921, 548 : 1228, 623 : 1536, 699 : 1843, 774 : 2150, 849 : 2457, 925 : 2764]$ for `CIFAR10` and $[256 : 156, 287 : 235, 317 : 313, 348 : 392, 378 : 470, 409 : 548, 439 : 627, 470 : 705]$ for `MNIST`, given the network architecture in Table 1 of Mariet & Sra (2016). These ratios correspond to 20% to 90% of the edges left for each node, as shown on the x-axis of Figure 1. These node-to-edge ratios are calculated based on the conversion equation in

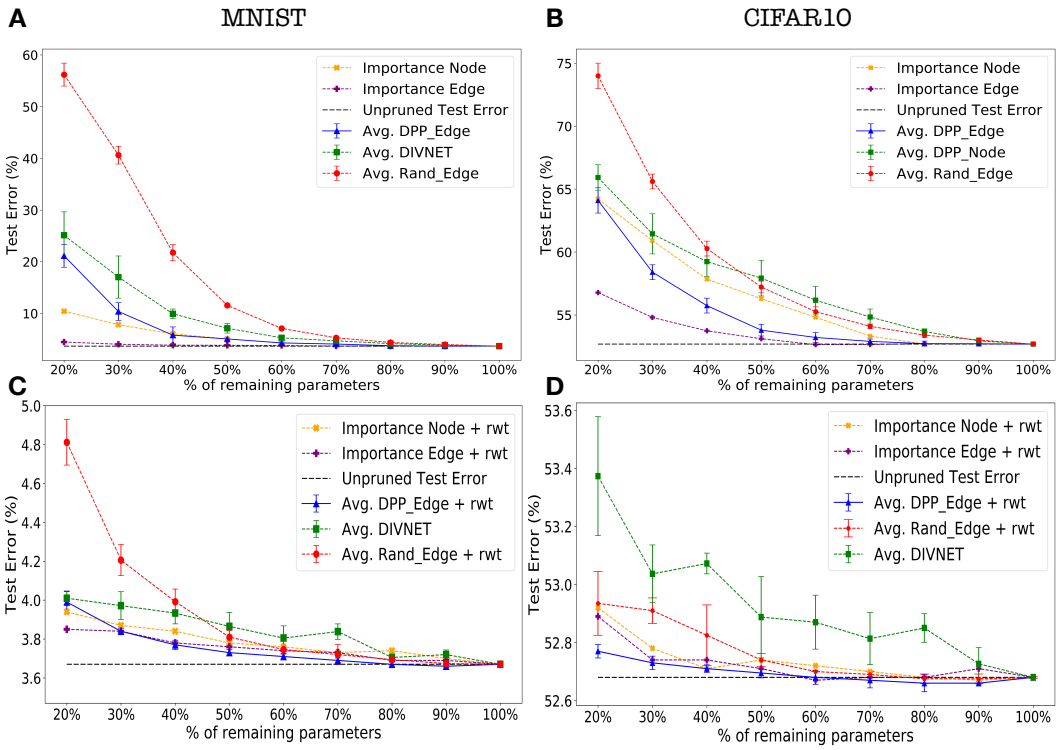

Figure 1: **(A-B)** Comparing pruning methods without reweighting on the MNIST (A) and CIFAR10 (B) dataset . **(C-D)** Comparing pruning methods with reweighting on the MNIST (C) and CIFAR10 (D) dataset. Horizontal axis represents the percentage of remaining parameters in 1^{st} layer after pruning. The vertical axis shows corresponding test error. It can be seen that when reweighting is not applied, importance edge pruning performs the best in both the datasets. However, after reweighting, DPP edge pruning beats all the pruning methods considered. Moreover, DPP edge also produces lower test error than the unpruned network for certain % (at 90% for MNIST and 70-90% for CIFAR10) without any retraining.

Section 4. We used $\beta = 10/|T|$ where $|T|$ is the size of the training dataset for all DPP node and edge kernel calculations on real data, following the choice of Mariet & Sra (2016).

4.2 RESULTS

We compare the test errors for the pruned networks with respective reweighting schemes for both datasets as in Figure 1. We observe that (1) When reweighting is not applied, importance edge pruning performs the best in both the datasets. (2) DPP edge pruned network significantly outperforms all other pruning methods when reweighting is added. This happens because, in the case of importance pruning, the greedy process of choosing the edges does not keep much room for improvement, whereas choosing the edges using DPP keeps enough space for improvement after reweighting. This shows the novelty of both DPP methods as well as the reweighting procedure. (3) Interestingly, we find that importance node pruning with reweighting performs better than DIVNET, which was not explored in Mariet & Sra (2016). (3) Furthermore, unlike other pruning methods, our sparse DPP edge pruned network generalizes better than the unpruned dense network for both the datasets (see at 90% for `MNIST` and 70-90% for `CIFAR10`) in Figure 1C and D). In literature, such sparse sub-networks of a dense network with better generalization ability is achieved via iterative pruning method Malach et al. (2020), which is time consuming. However, our approach gives rise to such sparse network without reinitializing and retraining but only reweighting adding to the novelty of our approach.

5 DISCUSSION AND FUTURE WORK

Our method falls under fast compression techniques that do not undergo retraining and outperforms competing methods in the same category on real data (Section 4). While DPP edge pruning is complementary to prior work, its novelty lies in theory, literature and neuroscience motivated definition. We consolidate the understanding of previous empirical findings and implications of observations in the brain from a theoretical perspective to propose a new pruning method. In future work the DPP edge pruning method can be tested on large network architectures and compared to other benchmark pruning methods that undergo retraining after pruning.

REFERENCES

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.

Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: a computational account. *Neural computation*, 10(7):1759–1777, 1998.

Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh PN Rao. *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.

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*, 2015.

Stephen José Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pp. 177–185, 1989.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pp. 164–171, 1993.

Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu. Reshaping deep neural network for fast decoding by node-pruning. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 245–249. IEEE, 2014.

Kumar Joag-Dev, Frank Proschan, et al. Negative association of random variables with applications. *The Annals of Statistics*, 11(1):286–295, 1983.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

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

Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 3(1), 2010.

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Zelda Mariet and Suvrit Sra. Diversity networks: Neural network compression using determinantal point processes. In *International Conference on Learning Representations*, 2016.