On Learning Wire-Length Efficient Neural Networks

Christopher G. Blake **Luyu Wang** **Giuseppe Castiglione**

Christopher Srinivasa **Marcus A. Brubaker**

Borealis AI
{chris.blake, luyu.wang, giuseppe.castiglione,
christopher.srinivasa, marcus.brubaker}@borealisai.com

Much recent work on computationally efficient image recognition has focused on minimizing the number of non-zero weights in the neural network [1–8]. This is a good energy measure for computation on a general purpose processor, however, as transistor sizes are approaching fundamental limitations, it may be more sensible to construct specialized neural network circuits [9, 10]. In specialized circuits, unlike in general purpose processors, energy consumption is often dominated by wiring length, and not number of memory accesses [11–15]. Moreover, wiring length is also optimized for in biological systems [16, 17], it correlates in practice to actual specialized circuit energy consumption [18], and is a fundamental engineering limitation for computation in the non-frictionless environments of the physical world [19, Chapter 8]. Inspired by this, in this paper we consider pruning algorithms to optimize for this wiring length measure using three techniques, each of which are used in different steps of the training-pruning process: weight-distance nested rank pruning, weight-distance based regularization, and layer-by-layer bipartite matching.

First, consider the neural network as a graph in the natural way, in which each neuron is a node and edges connect neurons associated with non-zero weights. A *placement* of nodes is a set of positions for each node of the neural network such that the distance between each node is at least 1 unit distance. The *wiring length* of an edge is the Manhattan distance between the nodes that it connects. The *energy* of the neural network is the total wiring length of all the edges that are not pruned.

The algorithm starts with a large neural network with a fixed number of nodes and some labeled training data. First, place the nodes in a *stretched-square grid layout*. In such a layout, we conceptually place the nodes of the neural network, layer by layer, within a square prism, which we call the optical channel. We let the width of the prism be just big enough to hold the nodes of the biggest layer in a single plane. Every other layer is placed consecutively on a square grid stretched out over the width of the optical channel, in a plane at unit distance away from the nodes of the previous layer. This ensures that all nodes are at least unit distance apart and that consecutive layers are adjacent. We list the three steps of our algorithm below, and provide a rough outline of the technique.

1. Train the neural network using stochastic gradient descent and *weight-distance regularization*. This involves using a cross-entropy cost function and a term defined as:

$$\Omega = \alpha \sum_i d_i^p |w_i|^2 \tag{1}$$

   In this notation, d_i denotes the length of the wire i and the w_i denotes it associated weight. Note that the $d_i s$ are fixed during this procedure, but the w_is are variables that are to be trained. The variables p and α are hyperparameters to search over. When p is non-zero, the regularization term is meant to bias the circuit to make long wires have lower weight.

2. Prune the edges of the neural network using a pruning criteria called *weight-distance nested rank pruning*. To do this, first sort the edges by weight and select all d_s fraction of lowest-weight edges. Then, sort these lowest-weight edges by distance, and prune a fixed number of the longest remaining wires. The parameter d_s is called distance sensitivity, when it is equal to the fraction of the wires that are pruned, this algorithm is equivalent to pure weight based pruning; when it is 1 this corresponds to pure distance-based pruning.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

3. After iterating between the previous two steps a sufficient number of times, apply *layer-by-layer bipartite matching* to further optimize the energy of the layouts. The algorithm uses the realization that finding the optimal permutation of nodes in one layer that minimizes the wiring length to the nodes of other layers assuming their positions are fixed is equivalent to the weighted bipartite matching problem, for which the Hungarian algorithm is polynomial-time and exact [20]. Apply this optimization algorithm layer by layer to the nodes of the pruned network.

Experiments

We run pruning experiments on a fully-connected neural network for MNIST, which contains two hidden layers of 300 and 100 units, respectively (this is the standard LeNet-300-100 architecture that has been widely studied in the pruning literature). We also try pruning the fully connected layers of a 10-layer convolutional network trained on the street-view house numbers dataset [21]. We show energy-accuracy curves for one setting of hyperparameters for each of these datasets in Figure 1.

In Tables 1 and 2 we show a subset of the results of a hyperparameter grid search for these two datasets. We record the accuracy and energy after each pruning iteration, and then for each set of hyperparameters choose the model with the lowest energy greater than some threshold accuracy. For each target accuracy we show the weight-based result (which is comparable to the technique of [3] and forms a baseline) and the results on the distance-based regularization technique. We found that nested rank pruning can perform better than pure weight based pruning, however distance-based regularization tends to outperform techniques that use nested-rank pruning, although sometimes distance-based regularization with nested-rank pruning performs best in the lower accuracy, low energy regime as can be seen in the right graph of Figure 1. In these tables we obtain a wide range of values at the highest accuracy (which we suspect is due to randomness in initial accuracy) but more consistency at the lower accuracies. For MNIST, our best performing set of hyperparameters results in a compression ratio of 1.64 percent at 98%, comparable to state-of-the art results for this initial architecture and dataset [22].

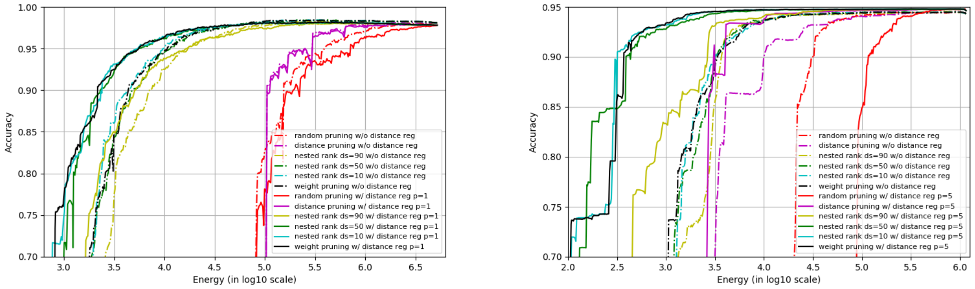

Figure 1: (left) Energy-accuracy curve for MNIST with $\alpha = 10^{-4}$ while varying distance sensitivity (ds) and p as labeled in legend. The black dashed line serves as a baseline weight-based technique; the other curves represent the introduction of nested-rank pruning and distance-based regularization. (right) Pruning of the fully connected layers of a 10-layer ConvNet for SVHN, with distance-based ($p = 5$ and $\alpha = 10^{-4}$, solid lines) or just L_2 regularization ($p = 0$ and $\alpha = 10^{-3}$, dash lines).

In Table 3 we apply the bipartite matching heuristic to the best performing network obtained using weight-based regularization and the best performing network using weight-distance based regularization for each target accuracy. Across both datasets the distance-based regularization outperforms weight-based regularization on average across four trials, in some cases by close to 70%.

Conclusion

In this paper we consider the novel problem of learning accurate neural networks that have low *total wiring length* because this corresponds to energy consumption in the fundamental limit. We introduce weight-distance regularization, nested rank pruning, and layer-by-layer bipartite matching and show through ablation studies that all of these algorithms are effective, and can even reach state-of-the-art compression ratios. The results suggests that these techniques may be worth the computational effort if the neural network is to be widely deployed, if significantly lower energy is worth the slight decrease in accuracy, or if the application is to be deployed on either a specialized circuit *or* general purpose processor.

α	p	pretrain accuracy		@98%	@97%	@95%	@90%
5×10^{-6}	0	97.9%	energy	117791 ± 28816	28412 ± 660	13594 ± 779	6831 ± 426
			edges	6831 ± 1649	1672 ± 43	787 ± 52	380 ± 25
	1	98.0%	energy	122322 ± 34712	28108 ± 2365	13986 ± 381	6964 ± 325
			edges	6815 ± 1929	1549 ± 130	766 ± 16	376 ± 17
	2	98.0%	energy	73878 ± 9034	22044 ± 1814	10373 ± 264	4893 ± 201
			edges	5087 ± 628	1518 ± 122	712 ± 17	337 ± 9
	3	98.1%	energy	107889 ± 12345	20474 ± 477	7370 ± 135	3613 ± 65
			edges	9920 ± 1053	2041 ± 53	744 ± 12	378 ± 5
	4	98.0%	energy	120307 ± 30421	14393 ± 696	6247 ± 292	2901 ± 149
			edges	13502 ± 3121	1895 ± 88	850 ± 42	412 ± 18
	5	98.0%	energy	205873 ± 68252	15744 ± 347	6348 ± 503	3004 ± 270
			edges	23414 ± 6433	2516 ± 42	1079 ± 82	530 ± 47
5×10^{-5}	0	97.8%	energy	93389 ± 5535	26777 ± 2105	13563 ± 401	6997 ± 202
			edges	5215 ± 307	1480 ± 113	737 ± 23	369 ± 14
	1	98.1%	energy	88241 ± 4613	20638 ± 2610	10141 ± 758	4559 ± 290
			edges	5971 ± 306	1430 ± 179	718 ± 53	328 ± 16
	2	98.1%	energy	47026 ± 3198	15354 ± 1800	7460 ± 775	3166 ± 244
			edges	4668 ± 282	1633 ± 185	813 ± 81	350 ± 29
	3	98.0%	energy	65236 ± 11137	14031 ± 702	7513 ± 658	3154 ± 373
			edges	7554 ± 1086	1934 ± 101	1075 ± 96	465 ± 55
	4	97.4%	energy	$-$	32258 ± 2307	12316 ± 693	5195 ± 588
			edges	$-$	4395 ± 283	1858 ± 96	802 ± 83
	5	96.1%	energy	$-$	$-$	16276 ± 1527	4742 ± 275
			edges	$-$	$-$	2307 ± 189	717 ± 42

Table 1: Results for MNIST task. Average and standard deviation of total wiring length (energy) and number of remaining weights (edges) over 4 random trials presented. Note that especially at lower target accuracies the distance-based techniques outperform the baseline, while number of remaining edges is comparable to weight based techniques (that is, when $p = 0$).

α	p	pretrain accuracy		@94%	@92%	@90%	@85%
5×10^{-5}	0	94.3%	energy	17759 ± 2107	6363 ± 62	4726 ± 267	3222 ± 203
			edges	1525 ± 187	538 ± 13	398 ± 23	271 ± 13
	1	93.5%	energy	$-$	5722 ± 379	3875 ± 260	2605 ± 241
			edges	$-$	558 ± 41	376 ± 26	249 ± 24
	2	93.1%	energy	$-$	3769 ± 461	2281 ± 159	1296 ± 63
			edges	$-$	561 ± 52	340 ± 19	200 ± 13
	3	94.1%	energy	7000 ± 794	1536 ± 95	1099 ± 71	717 ± 56
			edges	1616 ± 165	396 ± 22	290 ± 21	192 ± 13
	4	93.7%	energy	22718 ± 12008	1517 ± 133	1028 ± 34	718 ± 76
			edges	5215 ± 2259	516 ± 35	362 ± 6	262 ± 24
	5	94.4%	energy	3021 ± 217	748 ± 27	551 ± 43	416 ± 47
			edges	1163 ± 78	326 ± 15	242 ± 24	186 ± 28
5×10^{-4}	0	94.3%	energy	15155 ± 796	4975 ± 424	3215 ± 223	2113 ± 293
			edges	1294 ± 64	425 ± 40	275 ± 19	178 ± 25
	1	94.5%	energy	4883 ± 722	1853 ± 82	1130 ± 107	724 ± 15
			edges	666 ± 91	251 ± 17	151 ± 16	94 ± 3
	2	94.2%	energy	4329 ± 1360	1035 ± 106	651 ± 56	449 ± 56
			edges	1004 ± 279	270 ± 20	163 ± 14	115 ± 22
	3	94.0%	energy	5609 ± 337	616 ± 44	449 ± 26	350 ± 26
			edges	1683 ± 78	226 ± 12	163 ± 11	120 ± 11
	4	94.6%	energy	1616 ± 74	610 ± 32	520 ± 33	373 ± 69
			edges	673 ± 30	302 ± 14	265 ± 10	192 ± 35
	5	94.6%	energy	1444 ± 84	483 ± 12	348 ± 33	275 ± 52
			edges	654 ± 32	262 ± 7	194 ± 18	156 ± 29

Table 2: Average and standard deviation over four trials for Street View House Numbers task on both the wiring length metric (energy) and remaining edges metric (edges). We note that with the appropriate hyperparameter setting our algorithm outperforms the baseline weight based techniques (p=0) often on both the energy and number of remaining edges metric.

Task	p	α	bipartite matching	energy@97%	energy@95%	energy@90%
MNIST	0 (weight based)	5×10^{-4}	No	21403.6 ± 1092.5	10532.0 ± 204.4	4919.2 ± 348.2
	0 (weight based)	5×10^{-4}	Yes	8655.2 ± 545.8	3941.1 ± 119.0	1716.4 ± 134.3
	1 (distance based)	5×10^{-4}	No	12578.1 ± 586.7	5221.0 ± 272.0	2431.8 ± 230.2
	1(distance based)	5×10^{-4}	Yes	8561.9 ± 428.3	3253.7 ± 233.4	1460.7 ± 105.1
	p	α	bipartite matching	energy@92%	energy@90%	energy@85%
SVHN	0 (weight based)	5×10^{-3}	No	4957.8 ± 571.6	1963.7 ± 240.4	1034.2 ± 98.0
	0 (weight based)	5×10^{-3}	Yes	2929.8 ± 289.2	1111.9 ± 136.6	568.2 ± 47.1
SVHN	5 (distance based)	5×10^{-4}	No	483.3 ± 11.6	348.4 ± 33.3	275.1 ± 51.7
	5 (distance based)	5×10^{-4}	Yes	478.7 ± 10.8	343.6 ± 32.2	271.0 ± 53.2

Table 3: Results of applying the bipartite matching algorithm on the best performing weight-based pruning network and best performing distance-based regularization method before and after applying layer-by-layer bipartite matching. Average and standard deviation over 4 trials presented.

References

[1] Y. L. Cun, J. S. Denker, and S. A. Solla. Advances in neural information processing systems 2. chapter Optimal Brain Damage, pages 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[2] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[3] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[4] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[5] A. Torfi and R. A. Shirvani. Attention-based guided structured sparsity of deep neural networks. *CoRR*, abs/1802.09902, 2018.

[6] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016.

[7] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal. Wide compression: Tensor ring nets. *CoRR*, abs/1802.09052, 2018.

[8] A. Gordon, E. Eban, O. Nachum, B. Chen, T.-J. Yang, and E. Choi. Morphnet: Fast and simple resource-constrained structure learning of deep networks. *CoRR*, abs/1711.06798, 2017.

[9] O. Temam. Hardware neural networks: From inflated expectations to plateau of productivity. In *Federated Computing Research Conference*, FCRC '15, pages 4–, New York, NY, USA, 2015. ACM.

[10] Y. H. Chen, J. Emer, and V. Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM/IEEE Int. Symp. on Comp. Arch.*, pages 367–379, June 2016.

[11] C. D. Thompson. *A Complexity Theory for VLSI*. Ph.D. thesis, Carnegie-Mellon, 1980.

[12] P. Grover. Information friction and its implications on minimum energy required for communication. *IEEE Trans. Info. Theory*, 61(2), February 2015.

[13] C. G. Blake and F. R. Kschischang. Energy consumption of VLSI decoders. *IEEE Trans. Info. Theory*, 61(6):3185–3198, June 2015.

[14] R. A. Legenstein and W. Maass. Wire length as a circuit complexity measure. *Journal of Computer and System Sciences*, 70(1):53 – 72, 2005.

[15] K. Ganesan, P. Grover, and A. Goldsmith. How far are LDPC codes from fundamental limits on total power consumption? In *Allerton Conf. Commun., Control, and Comput.*, pages 671–678, Monticello, IL, 2012.

[16] D. Chklovskii and C. Stevens. Wiring optimization in the brain. *Adv. Neurol.*, 12:103–107, 01 1999.

[17] Schikorski T. Stevens C. F. Chklovskii, D. B. Wiring optimization in cortical circuits. *Neuron*, 34(3):341, 2002.

[18] K. Ganesan, P. Grover, and J. Rabaey. The power cost of over-designing codes. In *Proc. 2011 IEEE Workshop Signal Proc. Sys.*, pages 128–133, October 2011.

[19] C. G. Blake. *Energy Consumption of Error Control Coding Circuits*. PhD thesis, University of Toronto, Toronto, June 2017.

[20] H. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[22] J. Frankle and M. Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018.