Adapting Unstructured Sparsity Techniques for Structured Sparsity

Aditya Kusupati [1]

Abstract

Unstructured and structured sparsities provide unique advantages in resource-efficient sparse neural networks. Unstructured sparsity can assist in obtaining highly sparse and accurate models, while structured sparsity focuses mainly on enabling fast parallelizable inference on commodity hardware (e.g. GPUs). In the recent past, these distinctive advantages led to the divergence of the sub-fields leading to a disconnect. In this report, we propose and argue that most recent advances in unstructured sparsity can be adapted for inducing structured sparsity in deep neural networks. We also note the similarities between both these two sub-fields and document how the solutions from unstructured sparsity can be leveraged in solving the issues of structured sparsity. We also showcase the ease of adaptation by proposing $\mathrm{STR-BN}$ which is an application of the recently proposed STR method on batch normalization to induce structured sparsity via filter/neuron pruning. Code for $\mathrm{STR-BN}$ can be found at `https://github.com/RAIVNLab/STR-BN`.

1. Note

Deep neural networks have become the ubiquitous state-of-the-art machine learning solutions to a myriad of tasks spanning multiple domains. The prohibitively expensive resource demand of these models encouraged the community to look at deep neural networks through the lens of resource-efficient machine learning. Sparsity in deep neural networks is a promising research direction for achieving the real-world deployment of these models. Sparse neural networks have competitive prediction accuracy at a reduced resource utilization footprint i.e., model size, inference FLOPs, and working memory.

Sparsity in deep neural networks has been extensively studied in the literature and can be broadly classified as structured and unstructured sparsity based on the sparsity patterns. Unstructured sparsity does not take the structure of the model (e.g, channels, rank, neurons, etc.,) into account leading to seemingly random distribution of non-zero elements in the weight tensors. Gale et al. (2019), Savarese et al. (2019) and Kusupati et al. (2020) have extensively surveyed the current state-of-the-art unstructured sparsity methods. Unstructured sparsity methods span optimal brain damage (LeCun et al., 1990), global magnitude pruning (Han et al., 2016), gradual uniform layer-wise magnitude pruning (Zhu & Gupta, 2017), magnitude pruning with heuristic layer-wise budgets or reallocation of weights (Dettmers & Zettlemoyer, 2019; Evci et al., 2020) and learnable sparsity methods like l_1 regularization (Louizos et al., 2018) and STR (Kusupati et al., 2020). These unstructured sparsity methods attempted to solve the problems such as optimal pruning schedule, figuring out layer-wise budgets, and better heuristics to represent the importance of weights in model. In short, unstructured sparsity is induced by pruning out weights based on their importance (e.g. magnitude, gradient, momentum, hessian, etc.,).

Structured sparsity, on the other hand, takes structure into account and helps in the scaling of models on commodity hardware (e.g. GPUs, TPUs) which can support massive parallelization. Kuzmin et al. (2019) have surveyed on the structured compression and is a good read. Typically, structured sparsity includes methods which enforce sparse structure on parameter tensors like low-rank (Jaderberg et al., 2014; Kusupati et al., 2018), group sparsity through lasso (Wen et al., 2016), neuron/filter pruning using estimated importance of neuron/filter (Liu et al., 2019; Li et al., 2017; Liu et al., 2017; Luo et al., 2017; Hu et al., 2016). Most neuron/filter pruning methods relied on global threshold-based pruning to remove the neurons/filters in each layer, which could be sub-optimal and has similar problems as observed in unstructured pruning. To address this and figure out the optimal layer-wise sparsity budget, He et al. (2018) proposed a reinforcement learning based method to determine the sparsity ratio of each layer in terms of filters/neurons for a given resource budget. While AMC (He et al., 2018) focused on the right problem, a solution using reinforcement learning could be prohibitively expensive to obtain in the first place. Even though structured sparsity can leverage the speed-ups provided by massive parallelization, for a given

[1]University of Washington, USA. Correspondence to: Aditya Kusupati <kusupati@cs.washington.edu>.

parameter budget with high sparsity levels, unstructured sparsity methods typically result in more accurate models compared to structured sparsity methods. This can be noted by extensive comparison of sparse models across these subfields from the aforementioned works. However, structured sparsity (e.g. filter.neuron pruning) also tries to address the problem of memory/RAM constrained inference (Saha et al., 2020) as the activations of the deep neural networks tend to be much larger than the weights.

For the ease of exposition, this report only focuses on Convolutional Neural Networks (CNNs) but can be trivially extended to other models such as multi-layer perceptrons, recurrent neural networks, transformers, etc., The problem of good layer-wise sparsity budgets has plagued both unstructured and structured sparsity methods. Until recently, unstructured sparsity methods relied on global thresholding, uniform layer-wise sparsity budget, heuristic sparsity budgets like ERK to solve this problem. STR (Kusupati et al., 2020) along with other learnable sparsity methods proposed to directly learn the layer-wise sparsity budgets using backpropagation and SGD. This is much more efficient than the RL based method proposed in AMC (He et al., 2018), although AMC focuses on structured sparsity.

As noted above, most of the neuron/filter pruning methods prune them based on an estimated importance factor which could be a simple l_1 norm (Luo et al., 2017) of the filters or a proxy like the scaling factor in Batch Normalization (Liu et al., 2017). In a given layer of the neural network, the estimated importance scores of the filters form a single vector. When we look at the whole neural network through these importance scores, it is clear that structured sparsity can be modeled as an unstructured sparsity problem on these estimated importance vectors. Unstructured sparsity in each of these vectors corresponds to structured sparsity in the actual network resulting in a filter pruned model. This can also be extended to low-rank as shown by Kusupati et al. (2020) on FastGRNN (Kusupati et al., 2018; Dennis et al.).

Solving structured sparsity problems using unstructured sparsity methods on the estimated importance vectors helps the community leverage the advances in unstructured sparsity like heuristic sparsity budgets, learning sparsity budgets, stable pruning schedules among many others. It should be noted that inducing structured sparsity could be more unstable than unstructured sparsity hence the schedules might need a bit of modification to ensure stability in training. As observed above, the literature of these two sub-fields has been constantly diverging even when the methods are translatable/adaptable in both settings. For example, AMC (He et al., 2018) focused on RL based structured sparsity but the method should trivially translate to unstructured sparsity, albeit with more complexity.

In this technical report, we strongly encourage large-scale benchmarking, similar to (Gale et al., 2019), of unstructured sparsity techniques that can be trivially adapted to structured sparsity. The problems of sparsity ratios per-layer for structured sparsity can be attempted to solve using the heuristic or learnt budgets from unstructured sparsity literature. The incremental neuron/filter pruning schedules can be inspired by the gradual pruning schedule proposed in (Zhu & Gupta, 2017). Lastly, as exposited above, structured sparsity problems can be modeled as unstructured sparsity problems to leverage future advances.

2. $\mathrm{STR-BN}$

$\mathrm{STR-BN}$ is an example adaptation of unstructured sparsity methods to solve structured sparsity problems. Liu et al. (2017) proposed and showcased the effectiveness of using the scaling factor of the Batch Normalization (BN) (Ioffe & Szegedy, 2015) per neuron/channel as the estimated importance score for neuron/filter pruning. All the BN scaling factors are collected a global threshold is applied to obtain the desired sparsity. BN equation looks as follows: $\mathbf{Y} = \hat{\mathbf{X}} * \boldsymbol{\gamma} + \boldsymbol{\beta}$ where $\hat{\mathbf{X}}$ is the channel feature/activation tensor normalized with the running mean and variance, $\boldsymbol{\gamma}$ is the trainable scaling factor vector of BN and $\boldsymbol{\beta}$ is the trainable bias vector. Both $\boldsymbol{\gamma}, \boldsymbol{\beta}$ have the dimensional of output channels or the total number of filters in the layer making them correspond to each output channel or filter or neuron in the layer. Liu et al. (2017) used $\boldsymbol{\gamma}$ as the importance estimate per channel/neuron/filter. Motivated by this and the earlier discussion on modeling structured sparsity problems to solve using unstructured sparsity methods, we adapt a recently proposed learnable sparsity method for unstructured sparsity to support structured sparsity by operating on the BN parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$.

The proposed method, $\mathrm{STR-BN}$, reparameterizes the BN parameters (γ, β) using the Soft Threshold Reparameterization (STR). First, BN is modified as follows:

$$\mathbf{Y} = \boldsymbol{\gamma} * \hat{\mathbf{X}} + \boldsymbol{\beta}$$
$$\mathbf{Y} = \boldsymbol{\gamma} * (\hat{\mathbf{X}} + \frac{\boldsymbol{\beta}}{\boldsymbol{\gamma}})$$
$$\mathbf{Y} = \boldsymbol{\gamma} * (\hat{\mathbf{X}} + \hat{\boldsymbol{\beta}})$$

In this modification, both $\boldsymbol{\gamma}$ and $\hat{\boldsymbol{\beta}}$ are independent trainable vectors, * is a broadcast element-wise multiplication. We now apply STR on $\boldsymbol{\gamma}$ of each BN layer. From the reformulation, it is evident that making $\boldsymbol{\gamma}$ sparse will lead to sparse activations which in turn can be used to remove the filter that generated the zeroed out output channel. As STR induces sparsity gradually and facilitate learning layer-wise sparsity ratios it tries to address two important concerns of structured sparsity solutions. Owing to the sheer amount of compute required, the finer details of $\mathrm{STR-BN}$ such as the $g()$ function of STR are not fully explored. The code for STR – BN is at https://github.com/RAIVNLab/STR-BN, we invite and encourage any interested researchers to pursue it further as part of the suggested large-scale benchmarking of adapted unstructured sparsity methods for structured sparsity problems in machine learning models.

3. Conclusions

This technical report presented views on diverging sub-fields of sparsity (unstructured and structured) arguing that advances in either of them could potentially benefit the other. The report detailed how to adapt unstructured sparsity techniques to solve structured sparsity problems and presented STR – BN as an example of such adaptation. We highly encourage large-scale benchmarking of such adaptation and think it would benefit the whole community.

4. Acknowledgments

The author acknowledges support from his peers at the RAIVN Lab and is grateful to Ali Farhadi, Sham Kakade and Prateek Jain for their mentorship.

References

Dennis, D. K., Gaurkar, Y., Gopinath, S., Gupta, C., Jain, M., Kumar, A., Kusupati, A., Lovett, C., Patil, S. G., and Simhadri, H. V. EdgeML: Machine Learning for resource-constrained edge devices. URL https://github.com/Microsoft/EdgeML.

Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2020.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2016.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

Kusupati, A., Singh, M., Bhatia, K., Kumar, A., Jain, P., and Varma, M. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, pp. 9017–9028, 2018.

Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., and Farhadi, A. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, 2020.

Kuzmin, A., Nagel, M., Pitre, S., Pendyam, S., Blankevoort, T., and Welling, M. Taxonomy and evaluation of structured compression of convolutional neural networks. *arXiv preprint arXiv:1912.09802*, 2019.

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

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Saha, O., Kusupati, A., Simhadri, H. V., Varma, M., and Jain, P. Rnnpool: Efficient non-linear pooling for ram constrained inference. *arXiv preprint arXiv:2002.11921*, 2020.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2019.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.