Exploring the Design and Implementation of Pruning Techniques for Deep Neural Networks

Author: Andrea Bragagnolo
Supervisor: Prof. Marco Grangetto
Reviewers: Prof. Niculae Sebe, Prof. Alasdair Newson

Deep learning models have become increasingly complex and resource-hungry, making them difficult to deploy on devices with limited resources, such as smartphones or FPGAs. This has led to a shift towards client-server systems, where the trained model resides on a powerful server, and the client sends requests to the server for inference. However, this approach raises concerns about data privacy, internet connectivity, and latency. Researchers are exploring various techniques to reduce the computational requirements of deep learning models to address these issues, including re-designing network topologies, quantization, and pruning. These techniques aim to reduce the number of parameters, memory footprint, and computation required for inference, making it possible to deploy models directly on consumer devices. This work focuses on neural network pruning, providing several contributions to the literature.

First, we propose an unstructured pruning technique that removes the need for preliminary training of the model before applying the pruning step. Instead, our procedure, dubbed LOBSTER [5], employs a sensitivity-based regularization by exploiting the already available gradient of the loss function, avoiding additional derivative computations. This allows us to apply a joint train-and-pruned approach. We applied LOBSTER to standard computer vision tasks, such as image classification and segmentation, and compared it to state-of-the-art methods.

The unstructured nature of the sparsity introduced with LOBSTER can limit the practical applications of the pruned networks. We present a structured approach to confront this issue. SeReNe [7] is a method for learning sparse topologies with a structure by exploiting the concept of neural sensitivity as a regularizer. We define the sensitivity of a neuron as the variation of the network output with respect to the variation of the neuron's activity (i.e., the post-synaptic potential of the neuron), and the lower the sensitivity of a neuron, the less the network output is perturbed if the neuron output changes. Thanks to this sensitivity formulation, this procedure can drive all the neuron's parameters to zero, allowing learning of sparse network topologies with fewer neurons (fewer filters for convolutional layers). As a side benefit, smaller and denser architectures may also speed up network execution thanks to better use of cache locality and memory access pattern.

As seen throughout the thesis, modern pruning techniques can significantly reduce memory requirements and inference time, but they often have limited practical benefits when deployed on resource-constrained devices. Existing simplification methods have limitations, such as applying only to specific architectures or requiring special hardware or software. To ease the deployment of pruned neural networks, we propose "Simplify" [1], a PyTorch-compatible library that removes pruned neurons from a neural network, resulting in a smaller model that can be easily saved, shared, and used without special hardware or software. Simplify supports complex architectures like ResNet and DenseNet and can also be applied during training, reducing memory occupation and speeding up the training process. This library has proven beneficial to carry out the next experiments.

Moved by the interest in evaluating the applicability of neural network pruning, we carried out experiments to provide insight into the effect of pruning procedures on deployed models. In particular, we investigated the benefits of structured pruning approaches within the MPEG-7 Part 17 neural network compression pipeline. Evaluating how structured pruning can benefit quantization, entropy coding, and inference time [8, 2]. We experimentally show that, while the structured approach achieves a lower pruning ratio, it yields better end-to-end compression efficiency. Also, the network topology is easier to represent in memory once the network is decompressed, and the inference time is lower. Similarly, in collaboration with the Universitat Politècnica de València, we explored the deployment of pruned neural networks on FPGAs, proposing HLSinf [4]. Our evaluation demonstrates that quantized and pruned models can primarily benefit performance when combined with HLSinf. Specifically, results show that up to 90x speed up can be achieved on typical medical image-based applications using neural network models on FPGAs.

Other than just focusing on the structure of sparsification procedures, we studied the effect of one-shot and gradual pruning strategies. Furthermore, we will highlight some local properties of minima achieved using the two pruning strategies. To this end, we propose PSP-entropy [6], a measure of the state of ReLU-activated neurons, to be used as an analysis tool to better understand the obtained sparse network models. We have observed that one-shot strategies efficiently achieve moderate sparsity at a lower computational cost. However, there is a limit to the maximum achievable sparsity, which can be overcome using gradual pruning. Interestingly, the highly sparse architectures focus on a subset of sharp minima, which can generalize well, posing some questions about the potential sub-optimality of second-order optimization in such scenarios. This explains why one-shot strategies fail to recover the performance for high compression rates. More importantly, contrary to what could be expected, highly sparse, gradually pruned architectures can extract general features non-strictly correlated to the trained classes, making them unexpectedly, potentially, a good match for transfer-learning scenarios.

Finally, we tackle the problem of the resources required to train a neural network from a "pruning" perspective; in particular, we aim to slim the training process by selectively disabling the gradient computation for a subset of neurons. The proposed technique, NEq [3], allows us to find such neurons: we leverage the concept of neuronal equilibrium to freeze the gradient computation for neurons that no longer need updates. This reduces the time required for computing the gradients of the neural network during the backpropagation step. Various experiments on different tasks showed that NEq allows us to substantially reduce the number of operations needed to perform backpropagation.

References

[1] A. Bragagnolo and C. A. Barbano. Simplify: A python library for optimizing pruned neural networks. *SoftwareX*, 17:100907, 2022.

[2] A. Bragagnolo, E. Tartaglione, A. Fiandrotti, and M. Grangetto. On the role of structured pruning for neural network compression. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3527–3531, 2021.

[3] A. Bragagnolo, E. Tartaglione, and M. Grangetto. To update or not to update? neurons at equilibrium in deep models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22149–22160. Curran Associates, Inc., 2022.

[4] J. Flich, L. Medina, I. Catalán, C. Hernández, A. Bragagnolo, F. Auzanneau, and D. Briand. Efficient inference of image-based neural network models in reconfigurable systems with pruning and quantization. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2491–2495, 2022.

[5] E. Tartaglione, A. Bragagnolo, A. Fiandrotti, and M. Grangetto. Loss-based sensitivity regularization: towards deep sparse neural networks. *Neural Networks*, 146:230–237, 2022.

[6] E. Tartaglione, A. Bragagnolo, and M. Grangetto. Pruning artificial neural networks: a way to find well-generalizing, high-entropy sharp minima. *arXiv preprint arXiv:2004.14765*, 2020.

[7] E. Tartaglione, A. Bragagnolo, F. Odierna, A. Fiandrotti, and M. Grangetto. Serene: Sensitivity-based regularization of neurons for structured sparsity in neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021.

[8] E. Tartaglione, G. Nuzzarello, A. Bragagnolo, and M. Grangetto. Structured sparsity on embedded devices.