Lossless Compression of Deep Neural Networks

Thiago Serra [1] **Abhinav Kumar** [2] **Srikumar Ramalingam** [3]

Abstract

Deep neural networks have been successful in many predictive modeling tasks, such as image and language recognition, where large neural networks are often used to obtain good accuracy. Consequently, it is challenging to deploy these networks under limited computational resources, such as in mobile devices. In this work, we introduce an algorithm that removes units and layers of a neural network while not changing the output that is produced, which thus implies a lossless compression. This algorithm, which we denote as `LEO` (Lossless Expressiveness Optimization), relies on Mixed-Integer Linear Programming (MILP) to identify Rectifier Linear Units (ReLUs) with linear behavior over the input domain. By using ℓ_1 regularization to induce such behavior, we can benefit from training over a larger architecture than we would later use in the environment where the trained neural network is deployed.

Table 1. Compression of 2-hidden-layer rectifier networks trained on MNIST. Each line summarizes tests on 31 networks. Depending on how the network is trained, the higher incidence of stable units allows for more compression while preserving the trained network accuracy. For example, training with ℓ_1 regularization induces such stability and then inactive units can be removed. Interestingly, the small amount of regularization that improves accuracy during training also helps compressing the network later.

Layer width	ℓ_1 weight	Accuracy (%)	Compression (%)
25	0.001	95.76 ± 0.05	22 ± 1
25	0.0002	97.24 ± 0.02	8.3 ± 0.7
25	0	96.68 ± 0.03	0 ± 0
50	0.001	96.05 ± 0.04	29.4 ± 0.7
50	0.0002	97.81 ± 0.02	15.1 ± 0.6
50	0	97.62 ± 0.02	0 ± 0
100	0.0005	97.14 ± 0.02	30.8 ± 0.5
100	0.0001	98.14 ± 0.01	14.9 ± 0.4
100	0	98.00 ± 0.01	0 ± 0

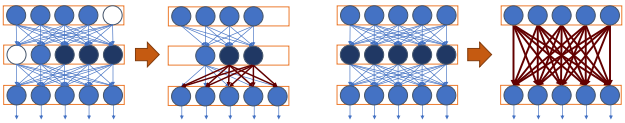

Figure 1. Examples of output-preserving neural network compression obtained with `LEO`. On the left, two units in white are stably inactive and three units indexed by set S in darker blue are stably active, where rank($\boldsymbol{W}_S^2$)=2. In such a case, we can remove the stably inactive units and merge the stably active units to produce the same input to the next layer using only two units. On the right, an entire layer is stably active. In such a case, we can fold the layer by directly connecting the layers before and after it. In both cases, the red arcs correspond to adjusted network coefficients.

[1]Bucknell University, Lewisburg, Pennsylvania, USA [2]Michigan State University, East Lansing, Michigan, USA [3]The University of Utah, Salt Lake City, USA. Correspondence to: Thiago Serra <thiago.serra@bucknell.edu>.

Extended abstract for the LXAI Workshop at the 37^{th} International Conference on Machine Learning (ICML 2020). The manuscript is available at https://arxiv.org/abs/2001.00218. The final paper will appear in *Proceedings of the 17^{th} International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2020).*