Supplementary
Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits

Robert Peharz [1] **Steven Lang** [2] **Antonio Vergari** [3] **Karl Stelzner** [2] **Alejandro Molina** [2] **Martin Trapp** [4]
Guy Van den Broeck [3] **Kristian Kersting** [2] **Zoubin Ghahramani** [5]

1. Comparison of Train Time, Inference Time and Memory Consumption – GPU

In Section 4.1 of the main paper, we compared training time and memory consumption for EiNets, LibSPN (Pronobis et al., 2017) and SPFlow (Molina et al., 2019), showing that EiNets scale much more gracefully than their competitors. The same holds true for *inference time*. Fig. 1 shows the results both for training time[1] $(\mathrm{sec}/\mathrm{epoch})$ and test time $(\mathrm{sec}/\mathrm{sample})$. Inference was done for a batch of 100 test samples for each model, i.e. the displayed inference time is $1/100$ of the evaluation time for the whole batch. Again, we see significant speedups for EiNets, of up to three orders of magnitude. We ran this set of experiments on a GeForce RTX 2080 Ti.

2. Comparison of Train Time, Inference Time and Memory Consumption – CPU

The improvements of EiNets stem mainly from the use of a few large einsum operations, where on the one hand we avoid much of the overhead of log-space computation, and on the other hand leverage the parallelism of GPUs. In order to disentangle these two different effects, we repeated the previous experiment on an *AMD Ryzen 7 3800X 8-Core processor*, restricted to a single core. The machine memory was $64\,\mathrm{GB}$. Restricting computation to a single core provides us with a rough idea how much of EiNet's improvements can be attributed to algorithmic changes, rather than to parallelism. In Fig. 2, we see the results of this experiment. We see that, overall, EiNets still dominate the other approaches both in runtime and memory consumption. The improvements are still roughly 1 order of magnitude in terms of speedup and up to 2 orders of magnitude in terms of memory consumption. Note that, like in the experiments using the GPU, LibSPN sometimes exhausted memory $(64\,\mathrm{GB})$.

[1]Eindhoven University of Technology [2]Technical University of Darmstadt [3]University of California, Los Angeles [4]Graz University of Technology [5]University of Cambridge, Uber AI Labs. Correspondence to: Robert Peharz <r.peharz@tue.nl>, Steven Lang <steven.lang.mz@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

[1]Same as in the main paper.

References

Molina, A., Vergari, A., Stelzner, K., Peharz, R., Subramani, P., Mauro, N. D., Poupart, P., and Kersting, K. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

Pronobis, A., Ranganath, A., and Rao, R. Libspn: A library for learning and inference with sum-product networks and tensorflow. In *Principled Approaches to Deep Learning Workshop*, 2017.

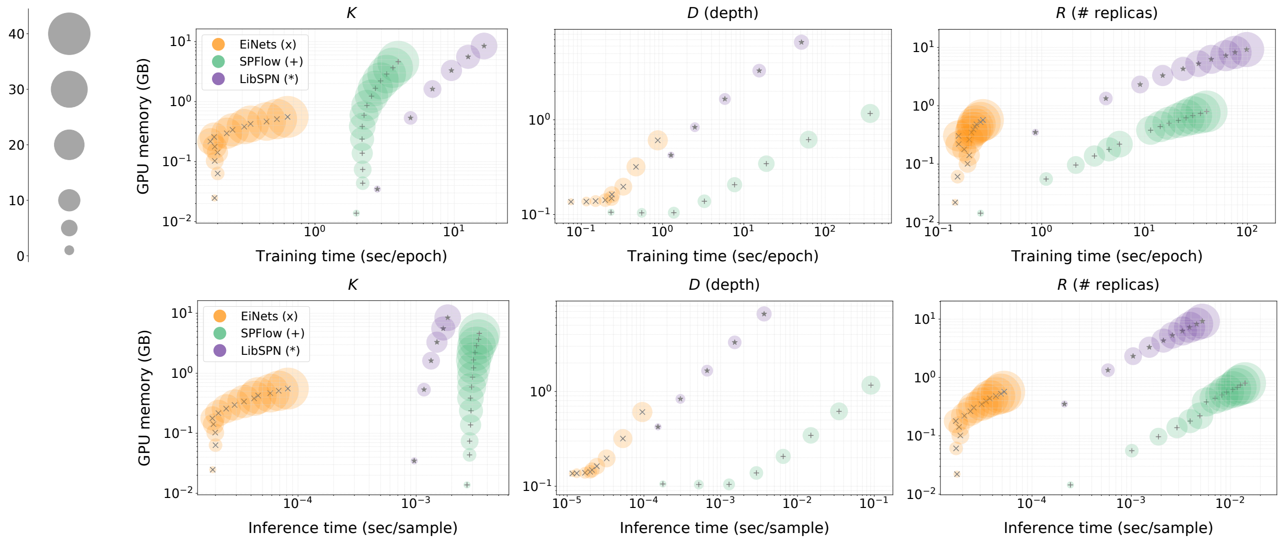

Figure 1. top: Illustration of training time and peak memory consumption of EiNets, SPFlow and LibSPN when training randomized binary PC trees, and varying hyper-parameters K (number of densities per sum/leaf), depth D, and number of replica R, respectively. The blob size directly corresponds to the respective hyper-parameter under change. The total number of parameters ranged within $10k - 9.4M$ (for varying K), $100k - 5.2M$ (for varying D), and $24k - 973k$ (for varying R). For LibSPN, some settings exhausted GPU memory and are therefore missing. bottom: Similar analysis as for top, but regarding inference time.
These experiments were run on a GeForce RTX 2080 Ti.

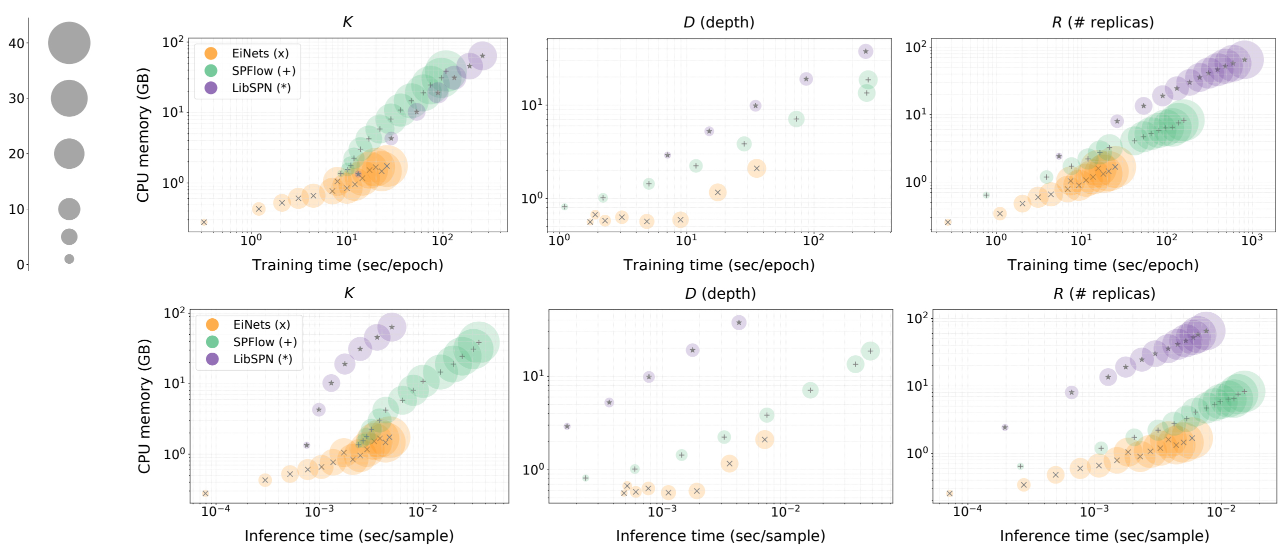

Figure 2. Similar results as for Fig. 1, but when run on a single core of an *AMD Ryzen 7 3800X 8-Core processor*, with 64 GB RAM.