Tuning the Learning of Circuit-Based Classifiers

Bernhard Gstrein
Johannes Kepler University Linz, Austria
Email: bernhard-gstrein@mailbox.org

***Abstract*—Modern predictive systems often come as big models with high computational costs. Circuit-based models could therefore be a step toward efficiency. As part of this author's Master's thesis, we present our efforts to construct classifiers that work internally with binary variables.**

I. Motivation

Neural networks trained using gradient descent work remarkably well. However, the sheer size of many models and the amount of computing required for training and inference are disadvantages. Devices with limited computing power cannot run large neural networks, so they must send data to a remote server. When there is no persistent internet connection, we cannot use these devices in conjunction with neural network-based artificial intelligence (AI). A small binary model, i.e., a circuit that requires little computing and disk space, might be a good step towards making AI more prevalent and reducing costs.

Existing work on circuit-based learning techniques that investigated lookup table (LUT) networks [1, 2] motivated us. We re-created various experiments and attempted to enhance the learning.

II. Related Work

The environmental impact of large predictive systems during training and inference is a growing concern [3]. Considerable research already exists to shrink neural architectures so they will fit on small electronic devices [4]. Most approaches to reducing model size into binary start with regular models and gradually transfer them to binary. We already started at binary.

III. Our Contributions

We investigated two schemes to build network-like architectures: LUTs and and-inverter graphs (AIGs). Both input and model are in binary. For simplicity, we also constrained the output class to be either 0 or 1.

The dataset that we used is a binarized version of MNIST [5] called Binary-MNIST. We evaluated the accuracy of various architectures on it, calculated the size they take up on disk, and discussed the computational power required to deploy these models.

Figure 1 summarizes our findings. We visualized testing accuracies and sizes for different models. The more a point lies on the top and the left, the better. For the meaning of the model names, we refer the reader to the author's thesis [6]. We took an existing approach of a binary model from [1], visible as "5@1024 8-LUT network", and were able to improve accuracy and reduce the size, visible as "(5@64)×1024 2-LUT advanced ensemble".

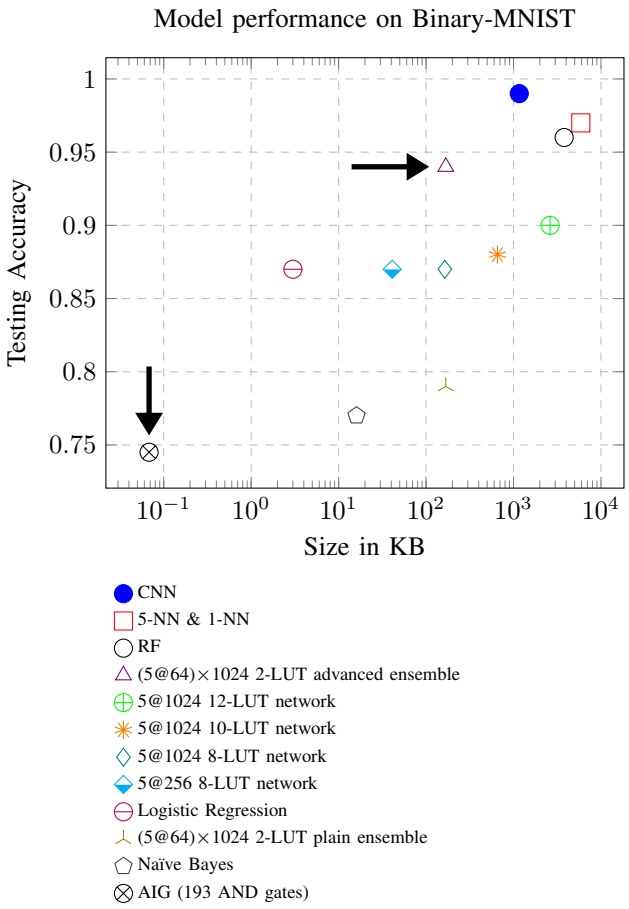
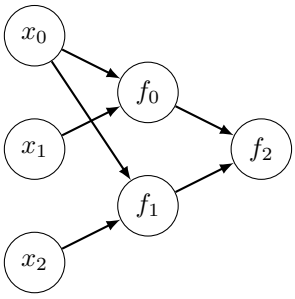


Fig. 1: Testing accuracy on Binary-MNIST and model size for various models. The arrows point out our main contributions.

In the lower-left corner, we can see the AIG. It is the weakest model in terms of accuracy. However, it is tiny, just 68 bytes, and easily translatable to a circuit on hardware. Thus, the AIG is a great candidate for implementation on small electronic devices.

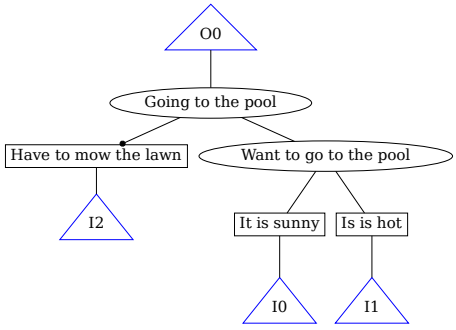
First described in [1], the LUT scheme features a specialized learning algorithm and outputs a classifier working entirely with binary values. Individual LUTs remember subsets of the training data. The LUT predictions are gathered in a vector and are the input to the next layer of LUTs. Stacking layers of LUTs enables performing complex tasks, while also keeping the arity, i.e., the number of bits taken as argument, low.



(a) Small 2-LUT network.

bit pattern	f_0	f_1	f_2
00	1	1	0
01	1	1	1
10	0	0	0
11	1	1	1

(b) Function table of the network in 2a.



(c) AIG classifier.

Fig. 2: Predictive systems that we have used in this thesis. The variables x_i are the inputs and either 0 or 1. The AIG also takes binary inputs. Each node represents the AND operation of its children.

Figure 2a visualizes a small 2-LUT network. We recreated one experiment from [1] and got the same results. To be viable for real-world ML use, we must improve the accuracy of LUT networks. Therefore, we took the idea of LUT networks further, where we took three main approaches:

- modifying existing LUT networks or the LUT network learning algorithm,
- enhancing the dataset using feature engineering, and
- combining many LUT networks of low arity (ensembling).

We were able to improve the accuracy, where we obtained the best results with an ensembling technique. An AdaBoost ensemble, which achieved a testing accuracy of 0.94, is better than any previous single LUT network. We could also reduce the size of the LUT networks significantly while retaining accuracy.

LUT networks with low arity can be easily improved by various tweaks to the learning algorithm and model architecture. For high-arity networks, however, the improvement often does not surpass 1-2%.

Concerning AIGs, we had a novel idea: utilize them as predictors. Usually, they are used as intermediate representations for circuits. Figure 2c visualizes a simple AIG classifier.

We devised an initialization scheme and a local search algorithm that we tried out by running experiments. The code for our experiments in C++ we wrote from scratch, making use of the AIGER library [7]. Choosing an architecture with just 192 AND-gates, the best local search run returned an AIG with an accuracy of 75%, which is significantly above chance.

The resulting AIGs were tiny, so their implementation on devices with little computing power is promising. However, the local search was slow.

IV. Future work

We must investigate other learning algorithms to see what works best for LUTs and AIGs. Until now, we have only done forward propagation, i.e., building layers sequentially. Learning via backpropagation would be a natural next step for LUTs.

Different AIG search methods could potentially be faster and increase the accuracy of the AIGs that result. Perhaps we could also use MaxSAT or other SAT-based methods.

We have yet to try out multi-class classification, which would be possible with LUT-based models and AIGs using a one-vs-one or one-vs-all approach similar to support-vector machines (SVMs) [8].

Acknowledgment

Special thanks to Martina Seidl and Armin Biere for acting as supervisors of this work.

References

[1] Satrajit Chatterjee. "Learning and memorization". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 755–763.

[2] Satrajit Chatterjee and Alan Mishchenko. "Circuit-Based Intrinsic Methods to Detect Overfitting". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1459–1468. URL: http://proceedings.mlr.press/v119/chatterjee20a.html.

[3] Roy Schwartz et al. "Green AI". In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[4] Di Liu et al. "Bringing AI to edge: From deep learning's perspective". In: *Neurocomputing* (2021).

[5] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[6] Bernhard Gstrein. *Tuning the Learning of Circuit-Based Classifiers*. eng. 2022. URL: https://resolver.obvsg.at/urn:nbn:at:at-ubl:1-51538.

[7] Armin Biere. *The AIGER And-Inverter Graph (AIG) Format Version 20071012*. Tech. rep. 07/1. Altenbergerstr. 69, 4040 Linz, Austria: Institute for Formal Models and Verification, Johannes Kepler University, 2007.

[8] Christopher M Bishop and Nasser M Nasrabadi. "Pattern recognition and machine learning". In: vol. 4. 4. Springer, 2006, pp. 338–339.