Weber State University

Master of Science in Computer Science

Thesis Proposal

Student name: Jarren Briscoe W#01257400

Supervisory Committee

Chair: Dr. Robert Ball E-mail: robertball@weber.edu
Member: Dr. Kyle Feuz E-mail: kylefeuz@weber.edu
Member: Dr. Brian Rague E-mail: brague@weber.edu

A Thesis Proposal for NN2Bool

1 Purpose

Describe or explain in detail the purpose or objective of the project (e.g. what is hoped to be gained or what problem is to be solved).

Neural networks (NNs) are a powerful blackbox machine learning technique. Many techniques have been done to extract rules or approximate neural nets [1, 2]. A viable, legacy whitebox structure that's formally verifiable is and-inverter graphs (AIGs) whose graphs are directed and acyclic. AIGs have two input nodes representing a logical conjunction, terminal nodes with variable names, and edges that may be marked to negate an input. While AND-inverter graphs are rarely structurally efficient for large circuits, they are effective with manipulating Boolean functions. And/or/inverter graph (AOIGs) and majority-inverter graphs (MIGs) are similar to AIGs. AOIGs have OR nodes in addition to what is available in AIGs. MIGs have majority nodes instead of the AND nodes in AIGs.

As defined in [3], "[r]ule extraction is an approach to reveal the hidden knowledge of the network and help to explain the process how neural network comes to a final decision." Using a rule extraction approach, I will be converting neural networks to a Boolean structure such as an AOIG or an MIG with some loss of accuracy. Let us call this conversion NN2Bool. NN2Bool

will be another tool to investigate neural networks to help understand the model's decisions for debugging and quality assurance.

2 Approach

Describe in one or two paragraphs the approach, technique or theory you intend to use to solve the problem.

In 2020, an article was published articulating how a group of researchers converted a neural network to and-inverter graphs. Shown in figure 1 are the three pipelines contrasted in [4].

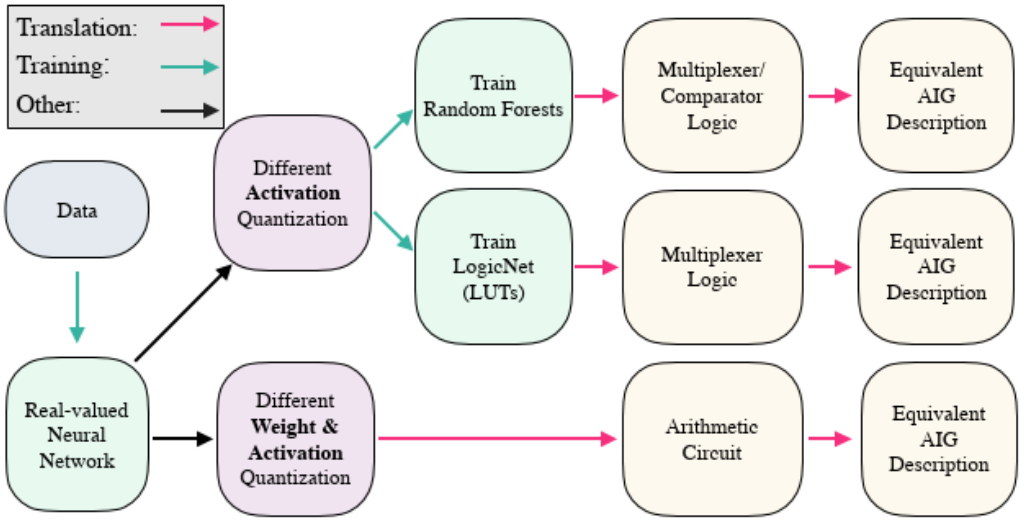

Figure 1: Pipelines contrasted in [4]

If I understand subsection 4.3 in [4] correctly, the researchers took each input for each activation node and trained a separate decision tree for each bit of each input. This technique seems like it would work, although inefficiently. Chatterjee created a network of lookup tables (LUTs) eliciting generalization, and each lookup table in the network was trained on the input from the layer prior and the final output. [4] referred to Chatterjee's network of LUTs as LogicNet. We could take the idea of global training from LogicNet to convert a neural network to an understandable structure by training a white-box learning algorithm on each activation node's inputs and the final output of the

neural network. In contrast, we could also employ a local training technique by training a white-box learning algorithm on each activation node's inputs and respective outputs. Every way mentioned above will create a network of white-box learning algorithms which then can be compiled to a Boolean structure.

MIGs (Majority-Inverter Graphs) were introduced in 2014 [5] with two basis operations: majority and inversion. A set of five primitive transformations completes a complete axiomatic system–allowing for easy traversal of the entire MIG representation space. On average, MIG optimization reduces the logic levels by 18% when compared to AIGs optimized by the state of the art ABC tool [6]. The reduced levels would make the graph easier to comprehend, however majority gates are not as intuitive as and/or gates. As proven in the article, MIG is a superset of AIG. One way to improve comprehension is to take the MIG and to find any nodes that can be converted to OR or AND nodes without increasing the total amount of nodes.

Figure 2 shows simple examples of converting AOIGs (And/Or/Inverter Graphs) to MIGs. Note that the Majority gate requires an odd fan-in and all trees are evaluated in a bottom-up fashion. These simple examples do not illustrate level reduction.

In [7], a neural network was trained using sigmoid activations then the sigmoid activations were transformed into step activations–allowing easier translation to an ordered binary decision diagram (OBDD). An OBDD is also called a "reduced ordered binary decision diagram" (ROBDD) or simply a "binary decision trees" BDD.

Taking ideas from this research, I plan on training a white-box algorithm on each activation and final output. Afterwards, this network of whitebox learning algorithms will be translated to Boolean logic. After converting this logic to a Boolean structure (such as MIG, AIG, AOIG), the Boolean structure will be reorganized and have redundancies removed using axioms described in [5] or with an open-source tool such as ABC [6].

Freely available software such as Python, Keras, Tensorflow, pandas, NumPy, and Theanos will be used throughout this project. A few open-source recommender systems which have a neural network will also be analyzed with NN2Bool.

Here are some open-source, neural-based recommender systems that are being considered:

- Neural Collaborative Filtering by Xiangnan He, Lizi Liao, Hanwang

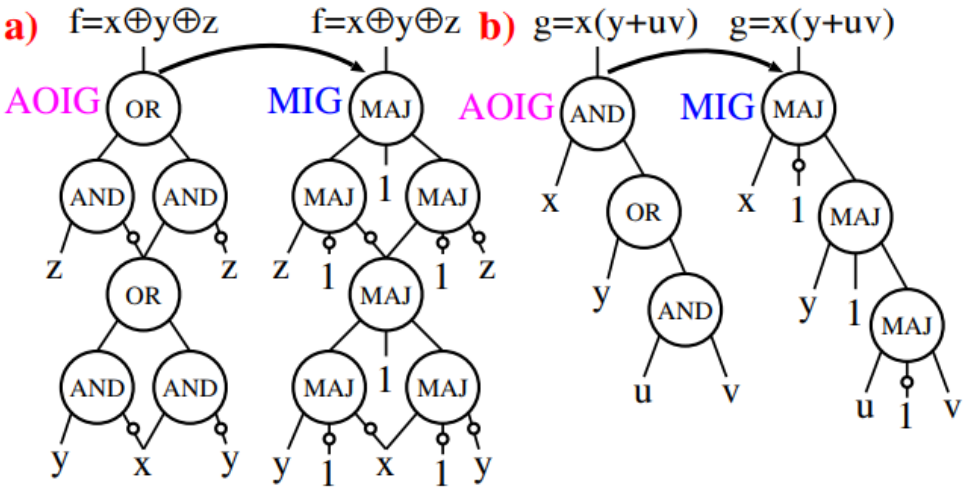

Fig. 1. Examples of MIG representations (right) for (a) $f = x \oplus y \oplus z$ and (b) $g = x(y+uv)$ derived by transposing their optimal AOIG representations (left). Complement attributes are represented by bubbles on the edges.

Figure 2: This image is "Fig 1" from "Majority-Inverter Graph: A novel Data-Structure and Algorithms for Efficient Logic Optimization" by Amaru et al. [5]

Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua. Source: Paper and Source Code [8]

- A tensorflow implementation of [8]. This repository also includes other recommendation systems.
- This repository represents three papers that consider feature-based recommendation [9], sequential recommendation [10], and social recommendation [11].

More open-source recommendation systems can be found here.

3 Research

Describe the research you have done that forms the basis for your approach, technique or theory. Include citations where appropriate.

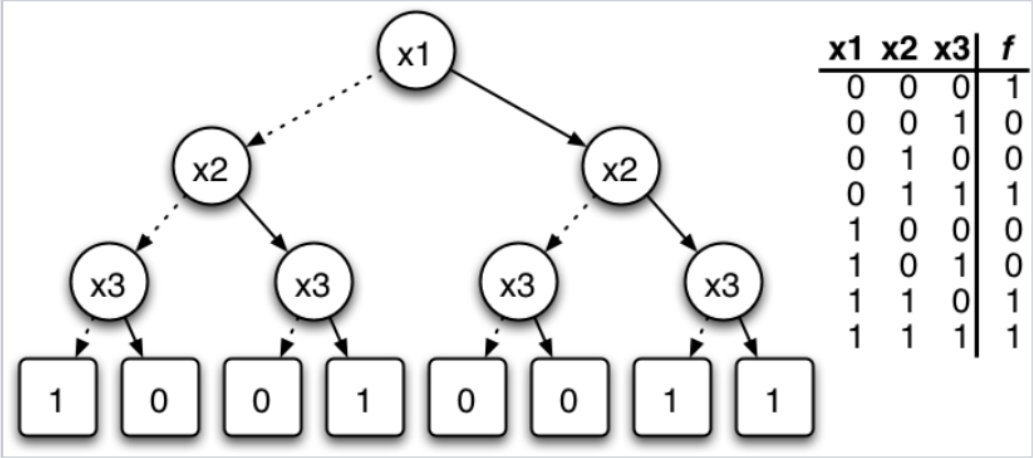

Figure 3: A binary decision tree. The solid edges assume the out-degree node is 1, and the dashed edges assume 0. This tree follows a top-down traversal.

Comprehension and Descartes Comprehension is a loaded term, yet the strongest form of comprehension can be expressed well by parts of Descartes' "Rules for the Direction of the Mind". Essentially, one must intuit each basic fact before using these basics in higher-level reasoning such as deduction. Since gaining an indubious understanding of each step takes time, it follows that the best way to help a user comprehend a trace from a Boolean structure is to reorganize the structure such that the trace has the least depth and the least amount of inputs at each level possible. Aspects of the more common but weaker form of comprehension include personality, trust, and probability–of which the philosophical and psychological backgrounds are beyond the scope of this paper. It is important that the user understand that he or she is comprehending the model which is an abstraction of the neural network. According to Descartes', believing in false relationships makes one less knowledgeable than if one did not know any relationship exists [12].

"Making Logic Learnable with Neural Networks". Brudermueller et al.

Introduction This paper combines the learning ability of neural nets with the interpretable and formally verifiable logic circuits. Brudermueller et

al. begin with a trained neural network then converts the neural network to random forests or multi-level look-up tables (LUTs), then to AND-Inverter graphs (AIGs). The researchers discovered that direct conversion from NN to AIGs does not create easily interpretable models and has a drastic loss of accuracy. The intermediate representation (IR) of random forests and LUTs elicit generalization in the final output. The LUTs used here is called LogicNet (a look-up table network).

There is very little research on logic operations being used to generalize. This paper points to "Learning and Memorization" (Chatterjee, 2018) as the only known example (Chatterjee used factoring to generalize). These simplified structures allow one to test sensitivity of the inputs.

Background In Boolean logic, AND is called a product and OR is called a sum, hence a sum-of-products means a set conjunctions (ands) joined with disjuntions (ors). The Boolean satisfiablity problem (SAT) asks if there exists a set of inputs (given some restraints) that invokes a true output. Public SAT solvers exist with O(n) complexity–where n equals the number of variables [13–15].

As mentioned above, Boolean functions can also be represented by AIGs. ABC can take AIGs as input and provides powerful transformations such as redundancy removal. ABC also includes a SAT solver called MiniSAT [6]. ABC seems like the best SAT solver to use for this project.

LUTs (look-up tables) save previously calculated results in an array-like structure. An N-bit LUT encodes a Boolean function with a fan-in of N with 2^N entries. LogicNet (Chatterjee, 2018) used LUTs in successive layers like a NN. Each LUT in a layer receives inputs from a few LUTs in the previous layer–these connections are chosen at random. This is a memorization process with noise as opposed to a backpropagation process. Each LUT in LogicNet is trained on the output of the entire network.

Related work

- Chatterjee and Mishcenko illustrated how circuit-based simulations can detect overfitting in [16].
- The researchers in [17] used a deep neural network to learn Boolean satisfiability as an alternative to SAT solvers.

- In [18], binarized weights and activations in neural networks were used as a precursor to hardware compilation.
- Murdoch et al. define interpretability as "the use of machine-learning models for the extraction of relevant knowledge about domain relationships contained in data." [19].

NN → Random forests → circuit → AIG Takes sets of activations from the trained NN and creates multiple training data sets to train multiple random forests on. General logic synthesis memorizes specific inputs which does not allow generalization–solution is to begin with a trained neural net. Using don't care minimization (some inputs do not affect the output) elicits an effect similar to K-nearest-neighbor interpolation based on cube expansion–this is their reason for using random forests. The don't care-based dependency elimination is a simple step to abstracting the formula. Example: $ab + a\neg b \rightarrow a$.

Why random forest: random forests explicitly selects the most important input variables–eliminating other dependencies.

NN → LogicNets → circuit →AIG Similar to the random forests pipeline, training data is created from the activations of the NN. Factored forms like multiple levels of LUTs induce generalization. Most straightforward circuit would be a sum of products. The sum of products (SOP) circuit lists every known possibility that creates a true output in conjunctions then if one of those conjunctions is true, the circuit outputs true. This is similar to listing sample data and is not enumerable nor generalizable.

Why Logicnet: Factorized LUTs generalize over simple LUTs by pulling common factors out of the training data (Chatterjee, 2018).

Both Both pipelines have lower complexity (AIG gate count) and improved accuracy compared to a direct NN → circuit transition.

Questions In 4.3.3 they say a random forest is trained on each bit of each feature as a binary classification problem. This seems very inefficient.

Prolog versus Boolean structures Prolog is based on Horn clauses, a subset of first-order predicate logic. While a single Horn clause is not

Turing complete, several Horn clauses can be arranged into a Turing machine. Therefore, Prolog is Turing complete.

Prolog is a form of logic programming which is also known as rule-based programming. Rule-based systems are more easily understandable than other methods since facts and rules are explicitly expressed.

As defined above, AIG and MIG are both Boolean structures. While I have found no research that attempts to convert neural networks into Prolog, there are some papers that convert neural networks to AIGs. Since MIGs can represent AIGs with less nodes, it would be interesting to convert NN2Bool's AIGs to MIGs to contrast. AIGs and MIGs have open-source graphing tools which add a visual aid to understanding the system. To the best of my knowledge, there are no tools with the same level of sophistication for Prolog available.

4 Criteria

Describe the Criteria that will be used to evaluate the project when it is completed. This section should contain sufficient detail so that there will be no misunderstanding between the student and the faculty about how the project will be graded.

1. Convert a trained neural net to an intermediate representation (IR). Let the IR be a network of decision trees.

    1.1. Inputs to the white-box algorithm will be the inputs to the activation node. Then two different tests will occur. One test will have the outputs be trained on the activation node's output (local training). The second test will be trained on the system's final output (global training).

2. Convert the IR to an AIG.

3. Use ABC to simplify and illustrate the AIG [6].

4. Contrast a simple Boolean data set, a complex data set with interpretable input, and a complex data set with uninterpretable input. Each data set will be trained on a neural network then the respective NN2Bool abstraction will be contrasted using accuracy, precision, recall, and F1 score.

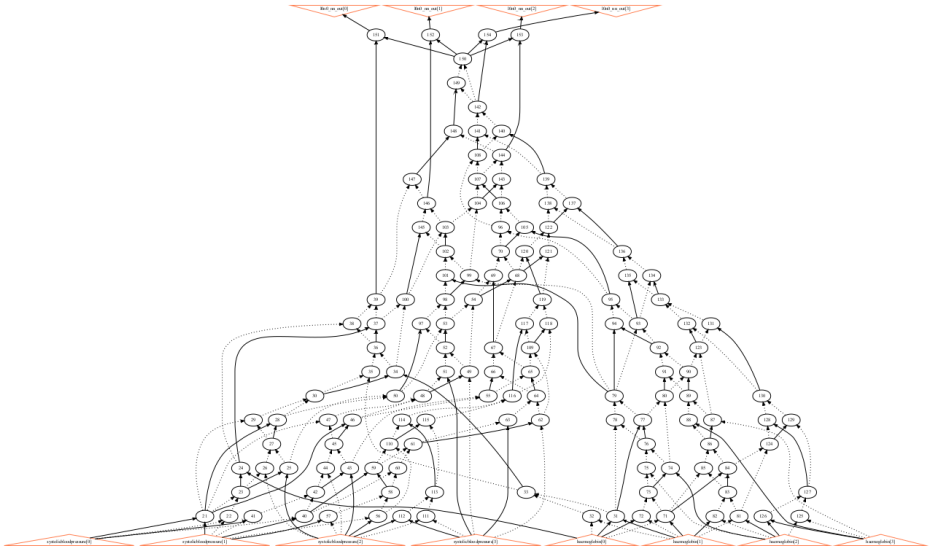

Figure 6. Example of the AIG that is derived from a simple logic module. The visualization was exported after creating the AIG with ABC (Brayton & Mishchenko, 2010).

Figure 4: A complex AIG with understandable input [4].

4.1. A simple boolean function. For example, take three binary inputs: green, yellow, and red. Then train on the function: "green and not red and not yellow".

4.2. Complex data with interpretable input such as the HIGGS data set. An example is given in figure 4.

4.3. Complex data with uninterpretable input such as the MNIST dataset.

5. Use NN2Bool to abstract a neural network from a recommendation system. The abstraction will be contrasted with the recommendation system's neural network using accuracy, precision, recall, and F1 score. The recommendation system may be taken from an open-source repository.

6. Compare NN2Bool on approximating neural networks that natively take Boolean input versus approximating neural networks that take

real-valued input. Both neural networks will have a categorical output. The real-valued input will be binarized. For example, if the real data (rd) has a range of between 0 and 1, then rd will be converted to 0 if rd is less than 0.5 else rd will become 1.

7. List the nodes taken in a path followed on the abstracted Boolean structure given user-provided input values.

Note: The data used for 4 and 6 may come from an open-source recommendation system. 4 only includes example datasets–the actual datasets tested may vary.

5 Committee Signatures

Chair _____ Date _____
Dr. Robert Ball

Member _____ Date _____
Dr. Kyle Feuz

Member _____ Date _____
Dr. Brian Rague

References

[1] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, 02 2018.

[2] R. Andrews, J. Diederich, and A. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems*, vol. 6, pp. 373–389, 12 1995.

[3] T. Hailesilassie, "Rule extraction algorithm for deep neural networks: A review," 09 2016.

[4] T. Brudermueller, D. Shung, L. Laine, A. Stanley, S. Laursen, H. Dalton, J. Ngu, M. Schultz, J. Stegmaier, and S. Krishnaswamy, "Making logic learnable with neural networks," 02 2020.

[5] L. Amarú, P. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2014.

[6] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," vol. 6174, pp. 24–40, 07 2010.

[7] W. Shi, A. Shih, A. Darwiche, and A. Choi, "On tractable representations of binary neural networks," 04 2020.

[8] X. He, L. Liao, and H. Zhang, "Neural collaborative filtering," *Proceedings of the 26th International Conference on World Wide Web*, 08 2017.

[9] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," pp. 1161–1170, 11 2019.

[10] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, and J. Tang, "Explainable knowledge graph-based recommendation via deep reinforcement learning," 06 2019.

[11] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 555–563, 2019.

[12] R. Descartes, *Rules for the Direction of the Mind*, vol. 1, p. 7–78. Cambridge University Press, 1985.

[13] N. Eén and N. Sörensson, "An extensible sat-solver," in *International conference on theory and applications of satisfiability testing*, pp. 502–518, Springer, 2003.

[14] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," in *Proceedings of the 38th annual Design Automation Conference*, pp. 530–535, 2001.

[15] E. Goldberg and Y. Novikov, "Berkmin: A fast and robust sat-solver," *Discrete Applied Mathematics*, vol. 155, no. 12, pp. 1549–1561, 2007.

[16] S. Chatterjee and A. Mishchenko, "Circuit-based intrinsic methods to detect overfitting," *arXiv preprint arXiv:1907.01991*, 2019.

[17] B. Bünz and M. Lamm, "Graph neural networks and boolean satisfiability," *arXiv preprint arXiv:1702.03592*, 2017.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, pp. 4107–4115, 2016.

[19] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: definitions, methods, and applications," *arXiv preprint arXiv:1901.04592*, 2019.