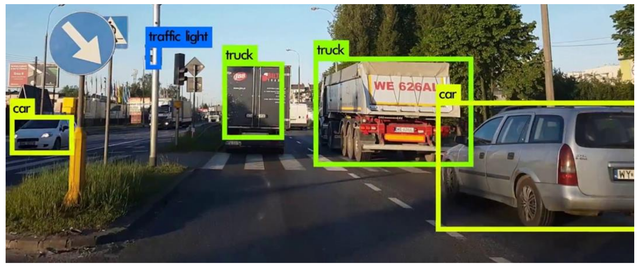
UR-70 Faster Inequivalence Testing Using Robustness

Abstract

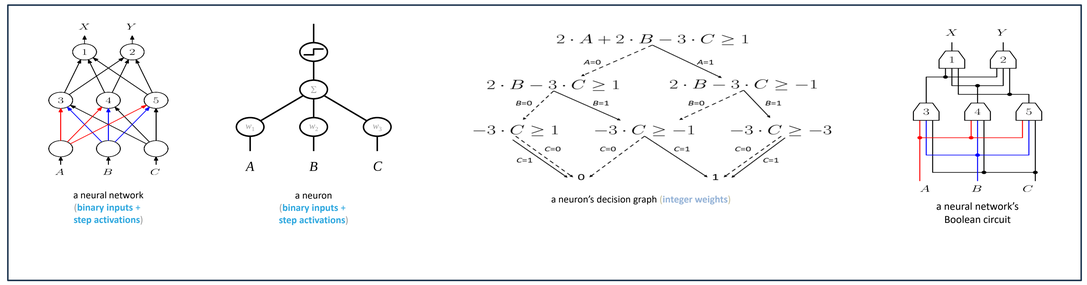
We propose a new method for quickly testing the inequivalence of two Boolean functions, when one function is represented as an ordered binary decision diagram (OBDD), and the other is represented in conjunctive normal form (CNF). Our approach is based on a notion of classifier robustness from the fields of explainable AI (XAI) and adversarial machine learning. In particular, we show that two Boolean functions that are very similar in terms of their truth values, can be very different in terms of their robustness, which in turn, provides a witness to their inequivalence. A more efficient approach to inequivalence testing has an impact on the development of more efficient model counters and knowledge compilers. In turn, such developments facilitate advances in explainable AI and adversarial ML.

Adversarial ML and Explainable AI



Over the past ten years, neural networks have drastically improved their ability to perform tasks like image classification and object detection. However, we've learned from adversarial machine learning that many of the classifications made by modern neural networks are in fact very fragile. For instance, strategically placing black and white stickers on a stop sign was enough to fool one deep neural network into thinking it was a 45 MPH speed limit sign. Examples like the stop sign show how important it is to be able to explain decisions made by artificial intelligence. These explanations, in turn, will provide a better understanding of important notions like classification robustness.

Compiling Neurons



As shown above, neurons in a neural network, provided they have binary inputs and step activation functions, can be compiled down to Boolean functions, shown here as a decision graph. The reason for this compilation is that Boolean functions generally have outputs that are much easier to explain relative to a neural network. If we're able to represent every neuron in a network as a Boolean function, then the whole network is equally representable as a Boolean function. However, this process of model compilation becomes more and more time consuming as the size of the neural network increases.

Leveraging Robustness

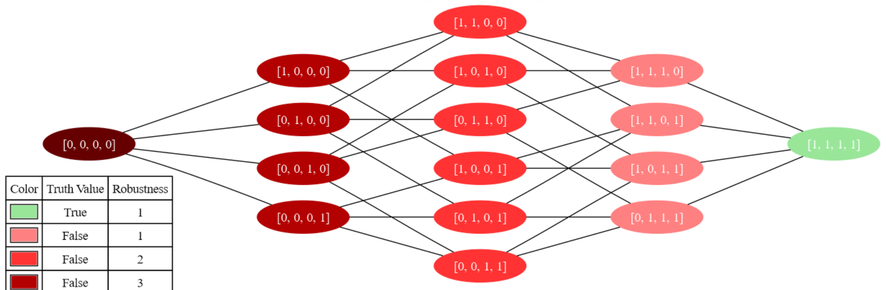
As we saw with the adversarial machine learning examples, many image classifications made by neural networks have a low robustness, causing them to be easily tricked with minimal changes to the input image. However, we can use that same notion of robustness to work in our favor. Although, instead of changes to the input image, robustness of a Boolean function is determined by how many changes to the input variables are needed to flip the function from True to False, or False to True.

$Theorem$: *For Boolean functions* f *and* g, *and variable instantiation* x.
If $f(x) = g(x)$, *but* $robustness_f(x) \neq robustness_g(x)$, f *and* g *are inequivalent.*

We ultimately concluded that even if two Boolean functions have equivalent truth values given an input, as long as their robustness is different, they're inequivalent.

Visualizing Robustness

$X_1 \wedge X_2 \wedge X_3 \wedge X_4$



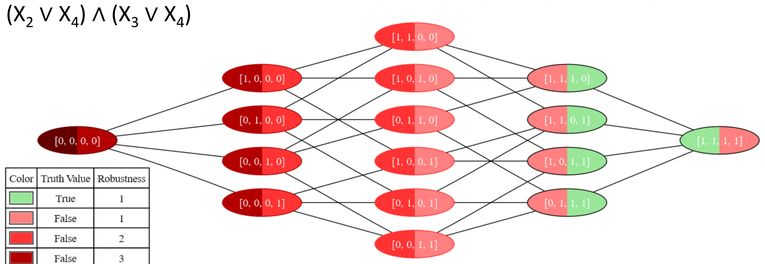
Color	Truth Value	Robustness
	True	1
	False	1
	False	2
	False	3
	False	4

To better understand robustness, consider the figure above, which depicts the robustness and truth values of a single Boolean function. Red nodes indicate that the function is False, green nodes indicate that it's True, and higher robustness at a node is shown by a darker shade of red/green. Note that the further to the left we traverse in this graph, the more variables we'd have to flip from "0" to "1" to get back to the green (True) node, this is the definition of an increase in robustness.

Inequivalence With Robustness

Left: $X_1 \wedge X_2 \wedge X_3 \wedge X_4$

Right: $(\neg X_1 \vee \neg X_2 \vee \neg X_3 \vee \neg X_4) \wedge (X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_1 \vee X_4) \wedge (X_2 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$



Color	Truth Value	Robustness
	True	1
	False	1
	False	2
	False	3
	False	4

By the above figure, we can now visualize how inequivalence can be found using robustness. To start, finding inequivalence the traditional way would require us to search the graph until we find a node where one half is red (False), while the other half is green (True). Though, with our theorem, we can find inequivalence at any node in this graph, as the robustness of the two functions is different at every node. Furthermore, the points of traditionally-found inequivalence cause a ripple effect on the robustness of their surrounding nodes, which is what causes the consistent differences in robustness.

Conclusion

This faster method of finding inequivalence will assist in the development of more efficient knowledge compilers and model counters, which will facilitate advances in explainable AI and adversarial ML. For future work, we plan to carry out a more exhaustive empirical study on our method's impact on the efficiency of existing knowledge compilers, with preliminary results showing a ~44% reduction in the number of equivalence tests required during compilation.

Acknowledgments

I'd like to thank my advisor, Dr. Choi, my two initial collaborators, Faye Le and Alex Drouillard, Professor Sharon Perry, Dr. Sanghoon Lee, and finally, KSU, for fostering the growth of undergraduate research at the university.

References

[Shi et al., 2020] Shi, W., Shih, A., Darwiche, A., and Choi, A. (2020). On tractable representations of binary neural networks. In Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR).

[Shih et al., 2018] Shih, A., Choi, A., and Darwiche, A. (2018). A symbolic approach to explaining Bayesian network classifiers. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI).

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Author: Emily Jackson – ejack150@students.kennesaw.edu
Advisor: Arthur Choi – achoi13@kennesaw.edu