Studies of machine learning algorithm application to digital circuit design and verification

Minsung Cho

Abstract - Machine learning is a sophisticated and statistical way of analyzing a pattern in a large and seemingly random dataset to predict the outcome and handle anomalies. It has been applied to many fields including medical, farming, and weather forecasting. In this paper, a couple of experiments bring machine learning to the electronics design and verification process.

I. METHODOLOGY

The main neural network machine learning algorithm that I selected is logistic regression. Simply put, logistic regression estimates the probability of the outcome from a set of independent variables. The library used is Tensorflow Keras. Keras is a deep learning library for deep neural network learning.
The program used for the circuit simulation is LTSpice. I first drew out the component circuit and turned them into a symbol. Then I plugged the symbol into the main circuit. Then I edit the netlist file and run numerous tests.

II. BACKGROUND

For the neural network, I chose a dense layer logistic regression sequential model. The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer.

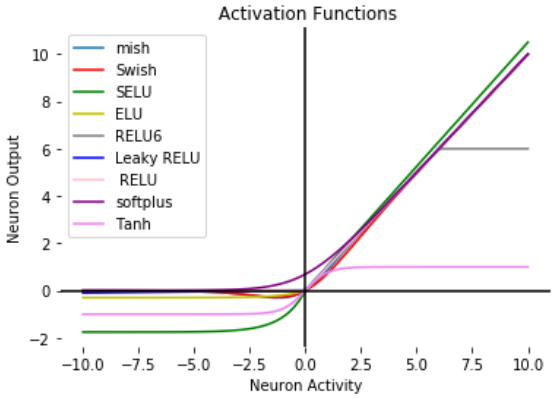

Fig. 1. Different activation functions are compared.

The activation function I chose is a rectified linear unit. A few advantages of the ReLU is that:

1. Only about 50% of hidden units are activated.
2. Fewer vanishing gradient problems compared to sigmoidal activation functions that saturate in both directions.
3. Scale-invariant.

It is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. A tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space.

The steps to take to make the network is:

1. Define network
2. Compile network
3. Fit network
4. Evaluate network
5. Make predictions with the model.

Deep learning has multiple layers of weights and perceptrons with hidden layers. The

optimization method used is Adam. Adam optimization algorithm is an extension to stochastic gradient descent. But it is different that stochastic gradient descent maintains a single learning rate(termed alpha) for all weight updates and the learning rate does not change during training while Adam calculates an exponential moving average of the gradient and the square gradient, and the parameters beta1 and beta2 controls the decay rates of these moving averages. The loss function used meant absolute error. It is the average of all absolute errors. The simple steps for it is as follows:

1. Find all of the absolute errors.
2. Add them all up.
3. Divide it by the number of errors.

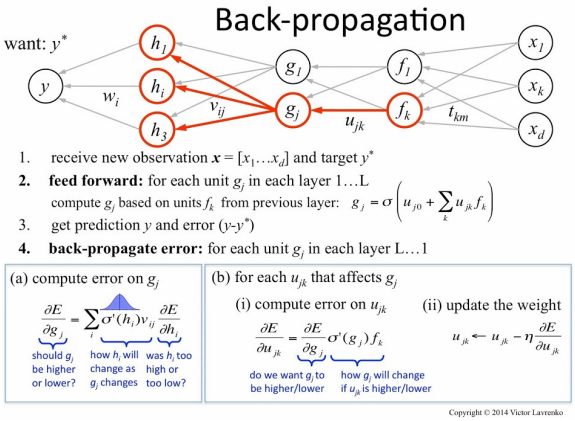

Fig. 2. Back propagation algorithm explained.

Fitting is adapting the weights on a trained dataset. Back propagation algorithm is used to fit the dataset. Back propagation computes the gradient of the loss function with respect to the weights of the neural network for a single input-output example. It works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

III. EXPERIMENT 1: CMOS INVERTER

First, the simple CMOS inverter design was used as an example for the purpose of being the proof of concept.

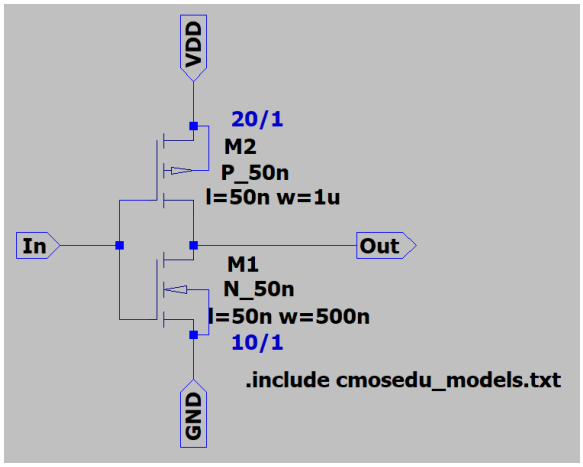

Fig. 3. The schematic of the inverter.

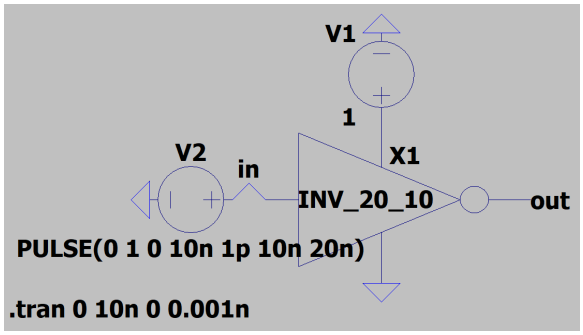

Fig. 4. The schematic of the main circuit under test.

The driven formula for the switching point for the inverter is

$$V_{SP} = \frac{\sqrt{\frac{\beta_n}{\beta_p}} * V_{THN} + (VDD - V_{THP})}{1 + \sqrt{\frac{\beta_n}{\beta_p}}}$$

Where β_n, $\beta_p = KP_n \frac{W_1}{L_1}$, $KP_p \frac{W_2}{L_2}$ where KP is the transconductance parameter and W and L are the widths and lengths of each CMOS transistors.

As you can see, the formula has a clear and linear relationship between the switching point and the log of NMOS size divided by PMOS

size. If both NMOS and PMOS lengths are the same, the switching point is simply going to depend on the width of PMOS divided by the width of NMOS.

Machine learning, let alone deep learning, is not necessary if the function is already driven. In Particular, the inverter is the simplest form of a digital circuit. However, as the circuits get more complex, the function gets more and more convoluted. Also, there are various factors that are omitted in the formula which used to be negligible in an older process but which get more and more significant as the size factor of the process gets smaller. Therefore, the purpose of the inverter example is the proof of concept to apply machine learning to simulate and optimize the desired output of more complex digital/analog circuits.

10,000 simulation data was collected by incrementing the lengths and widths of NMOS and PMOS shown in figure 1 by the C5 process increment of 50n(m). The initial values are as follows:

- NMOS length = 50nm, width = 500 nm
- PMOS length = 50nm, w = 1000nm

The drain power voltage was kept at 1V and the input voltage was set to be a linear sweep from 0 to 1V in 10nm rise time. The output was set to be the switching point voltage, which is when the input voltage and the output voltage are equal to each other. This value is a floating point.

The data was split into 80% train data and 20% test data. Within the train data, 80% was used for validation.

And the deep learning model is set to have 2 hidden dense layers of 64 units and a single output dense layer.

The optimizer alpha rate was set as 0.001. The training was done for a total of 100 epochs.

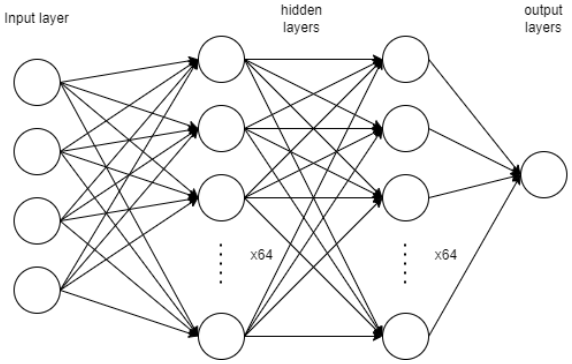

Fig. 5. The diagram of the custom model.

The result of the training and the reduction of the loss function is shown below.

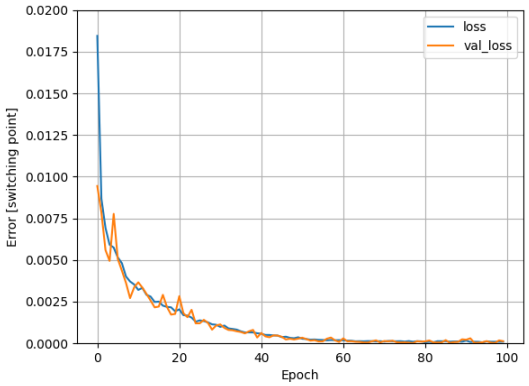

Fig. 6. The loss and the validation loss decreases as epoch iterates.

The final error figure was 0.000099 for loss and 0.000131 for the validation loss.

Finally, there are some unforeseen problems that can be observed from the experiment. First, the input data is quantized and therefore not floating point. Neural network performs optimally when floating point data is fed and trained. Second, beyond a certain point of the length and the width, incrementation is not necessary. The smaller the transistors the better. Therefore, increasing the lengths and the widths of the devices to generate a large number of data is not necessary. And third, many other aspects were

ignored, such as VDD noise and variation, rise and fall time, and switching point of the rising and falling edges. To improve the justification of neural network application, the next experiment was done with a device called a schmitt trigger.

IV. EXPERIMENT 3: CMOS SCHMITT TRIGGER

A Schmitt trigger is a circuit useful in generating clean pulses from a noisy input signal or in the design of oscillator circuits. Compared to the inverter, Schmitt trigger has a steeper transition region and hysteresis. It can have two different switching points for falling edge and rising edge.

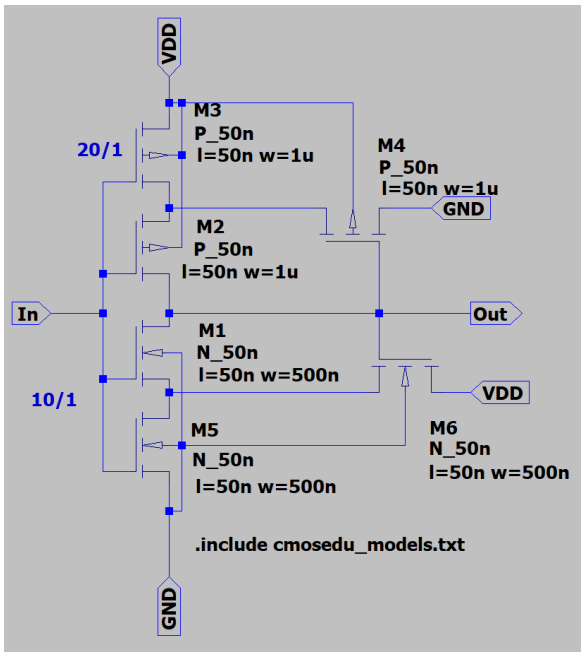

Fig. 7. Schematic of Schmitt trigger.

The Schmitt trigger also has a mathematically driven formula for its high and low switching points.

For high switching point:

$$\frac{\beta_1}{\beta_3} = \frac{W_1 L_3}{L_1 W_3} = \left[\frac{VDD - V_{SP}}{V_{SPH} - V_{THN}}\right]^2$$

And for low switching point:

$$\frac{\beta_5}{\beta_6} = \frac{W_5 L_6}{L_5 W_6} = \left[\frac{V_{SP}}{VDD - V_{SPL} - V_{THP}}\right]^2$$

A couple of things to notice here is that it is a slightly more complex equation. However, the temperature is not in the equation, which I will also introduce.

For the schmitt trigger simulation, all lengths of the CMOS were fixed to 50nm. The NMOS M1 and M5 and PMOS M2 and M3 widths were also fixed to 500 nm and 1um each. The only incrementing variable will be the width of PMOS M4 and NMOS M6. The M6 width was swept from 50nm to 500 nm(10 steps) and the M4 width was swept from 50nm to 1000nm(20 steps). Temperature was also added randomly from the interval between -60 to 140 degree celsius(5 steps). On top of that, VDD will have a 10% random noise between 0.9 and 1.1V. Total size of the dataset will be 10 * 20 * 5 = 1000. I have collected 972 data with a complete randomization of all the figures this time instead of trying to include every case for the incremental variables.

The input variables are:

- NMOS M6 width(50 nm increment, quantized integers).
- PMOS M4 width(50 nm increment, quantized integers).
- Temperature(floating point).
- VDD(floating point).

And the output variables are:

- Falling edge switching point(floating point).
- And rising edge switching point(floating point).

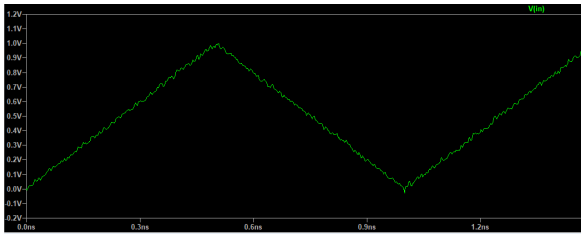

Fig. 8. Generation of the random wave with noise.

As you can see in the figure 6, the input triangular wave has a noise attached. Similarly, the drain voltage will have a 5% margin of noise introduced.

For the data collected before training, looking at the correlation heatmap shows that there is a slight but noticeable positive correlation between NMOS width and the high switching point. It is interesting to note that VDD was not a big factor despite it being in the formula for the high switching point voltage. Also, it can be seen that the PMOS width and VDD are negatively related to the low switching point voltage.

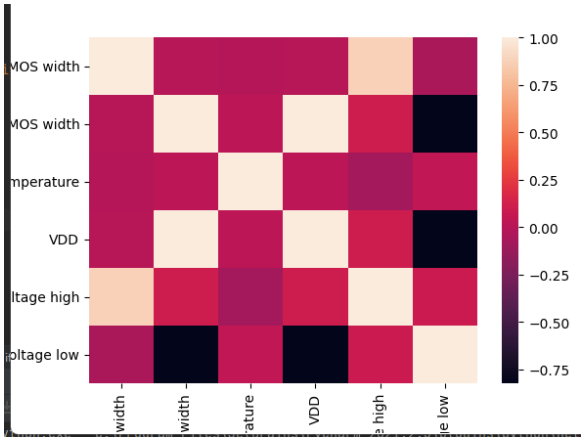

Fig. 9. Heatmap correlation of the variables.

And here is the pair plot of the variables for a visual regression. Even though the scatterplots show some trends, it can be noticed that the trend is not completely linear.

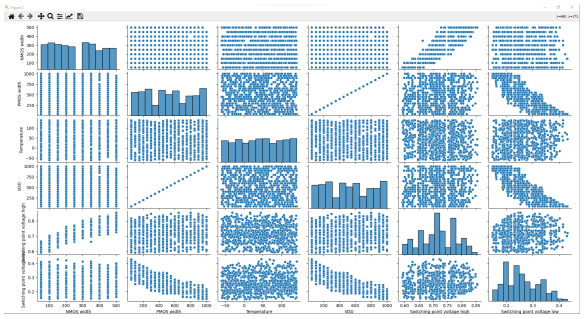

Fig. 10. Pairplot of the variables.

The deep learning model for Schmitt trigger has 4-unit input layer, 4 16-unit hidden layers, and 2-unit output layer. Same as the previous model is the usage of Adam optimizer. The reason for the reduction of the unit count for each and increase of the number of layers is to introduce nonlinearity. Generally the more units the better diversity but since I have doubled the layer size, for computational efficiency, I have decreased the unit size for 16 each layer.

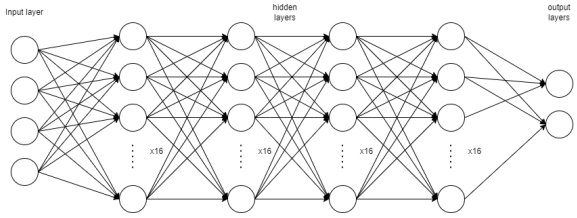

Fig. 11. Deep learning model for schmitt trigger

What is different from the previous model is that the error function I selected is root mean square error(RMSE) instead of mean average error(MAE). The formula has a proven advantage over MAE by other experiments and papers.

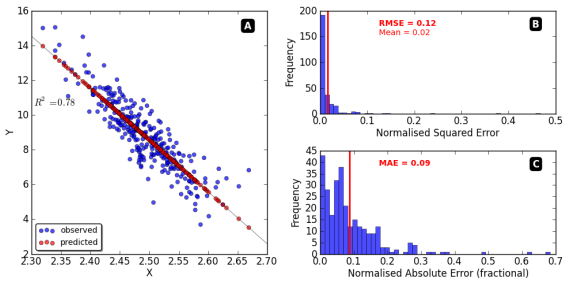

Fig. 12. RMSE vs MAE error frequency

The result of a 100 epoch yielded a high switching point error rate of 0.033 and a low switching point error rate of 0.036. It was able to predict both points accurately.

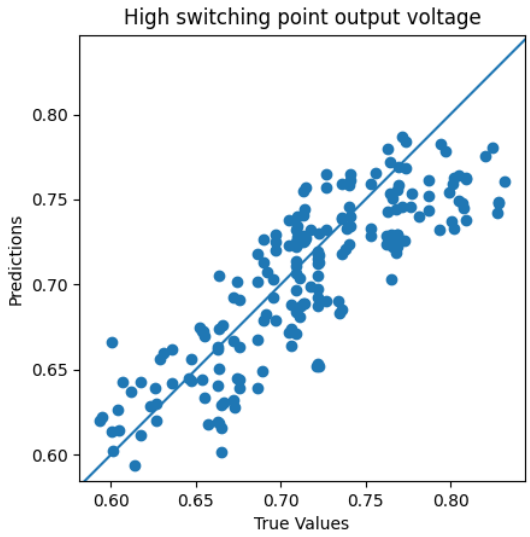

Fig. 13. High switching point prediction vs true value.

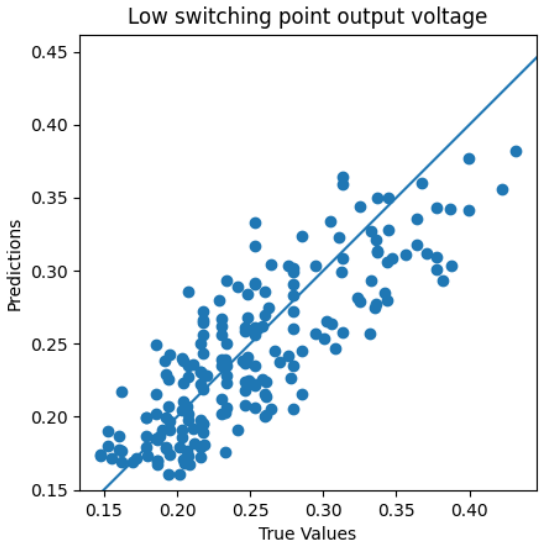

Fig. 14. Low switching point prediction vs true value.

One thing to notice from the RMSE and loss function is that because of the multiple layers, it was able to be very accurate within 20 epochs.

Therefore, even 16 units and 4 layers were computationally redundant. Building a model with hyperparameters will always be a very intricate part where a plug-and-play to achieve the balance between the performance and efficiency.

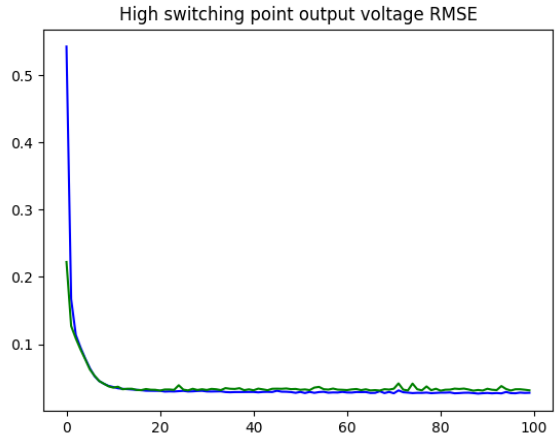

Fig. 15. High switching point RMSE

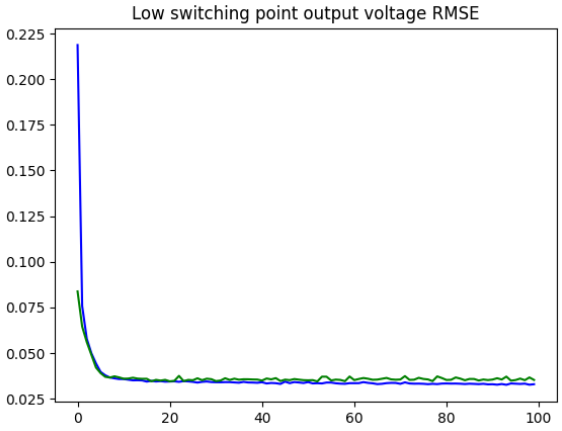

Fig. 16. Low switching point RMSE

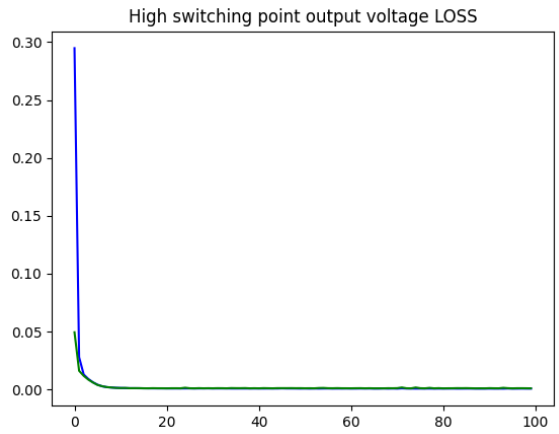

Fig. 17. High switching point loss

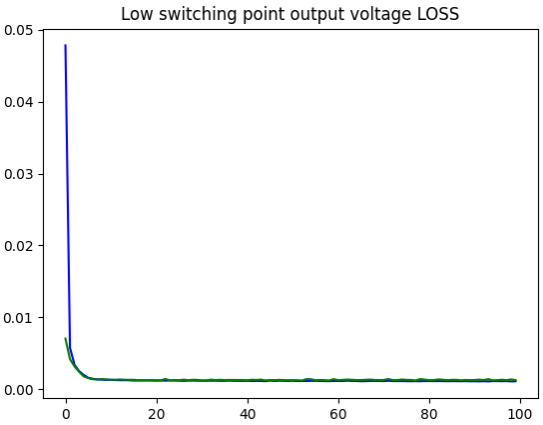

Fig. 18. Low switching voltage loss

Here is the numerical data after the training:

- loss: 0.002378956414759159
- SP_high_loss: 0.0011043667327612638
- SP_low_loss: 0.0012745896819978952
- SP_high_err: 0.03323201462626457
- SP_low_err: 0.03570139780640602

The purpose of this experiment is to show that more complex circuits can be verified and designed with machine learning. A circuit with thousands of variables and hundreds of outputs can be verified and analyzed the same way as the simplest form of the digital circuit can be analyzed. When a more sophisticated design tool such as Cadence Virtuoso or OrCAD is utilized, there will be more variables and unknown conditions on the table to be experimented on, drastically changing the traditional method of semi-plug-and-play.

V. FUTURE WORK

There is a great interest in reducing the layout size of any electronics design to reduce the total cost and delay of the logic modules. One of the ways to do this is to use logic synthesis. However, there is a special type of a very customizable circuit called a threshold logic gate. It has a multiple input and multiple output that uses the weights for each inputs rather than complex logic gate combinations.

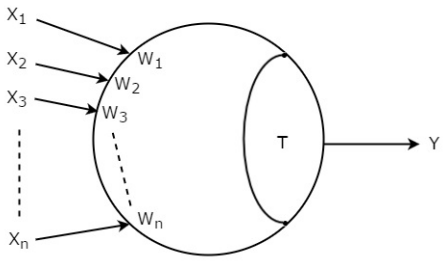

Fig. 19. Threshold logic gate diagram.

The purpose of threshold logic gate is to reduce the layout size of a complex logic gate by using weights instead of many logic gates introduced by synthesis tools such as Karnaugh map.

What is interesting about this concept is that it is very similar to how a perceptron works. Just like the perceptron, threshold gate's output will be adjusted by the weight of each input and their sums. The threshold gate will have a lot of complex variables depending on the number of inputs, and as deep as my research, there is no adequate equation to design the weights. Machine learning can be a great tool for this case.

Below is the calculation of the number of functions of threshold gate versus the boolean functions. As the number of inputs increases, the complexity of the threshold gate is comparatively lower than the boolean logic.

n	NUMBER OF THRESHOLD FUNCTIONS B_n	TOTAL NUMBER OF BOOLEAN FUNCTIONS $\left(2^{2^n}\right)$
1	4	4
2	14	16
3	104	256
4	1,882	65,536
5	94,572	4.3×10^9
6	15,028,134	1.8×10^{19}
7	8,378,070,864	3.4×10^{38}
8	17,561,539,552,946	1.16×10^{77}

Fig. 20. Comparison of threshold functions vs boolean functions.

VI. CONCLUSION

In this paper, I have demonstrated the application of deep learning algorithms to verify and design rather simple circuits. The next step to take is to find a more complex circuit that can benefit from the algorithms that do not have a driven and simple formula for its functionality and dependencies. The other route is to work on a specific circuit such as the threshold logic gate to develop a new kind of optimized logic module that will be a competition to the traditional gates.