SAT based Attacks on ML Models

Saaketh Gunti : 190101080

Department of Computer Science and Engineering,
IIT Guwahati

April 26, 2023

Model Extraction Problem

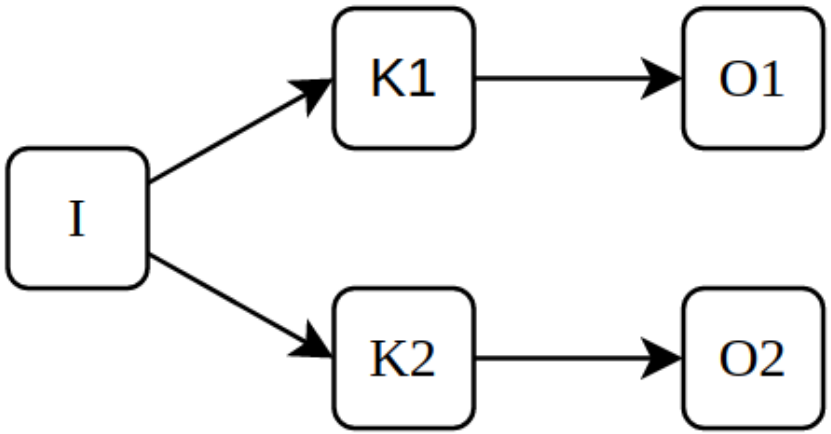
- High performance ML models are kept secret for multiple reasons.
- Only Black box access and Structure of the model are shared.
- Extracting the inner details of a trained ML model given the structure and black box access to it.
- There are two types of Model Extraction Attacks:
  1. Task-Accuracy Extraction
  2. Fidelity Extraction

Problem Statement

- Given the structure and oracle access to the Neural network, we have to find a combination of parameters such that the network built gives the exact same result as the oracle for all valid input combinations using SAT-based attack.
- Formally, let $f : X \rightarrow Y$ represent the Neural Network, we want to find a function $g : X \rightarrow Y$, such that for every input vector $x \in X$, $f(x) = g(x)$.

Distinguishing Input Pattern (DIP)

$$S = \{K1, K2, ..., Kn\}$$



$$g(I, K1) = O1$$

$$g(I, K2) = O2$$

$$O1 \; != \; O2$$

Figure: DIP

SMT Attack

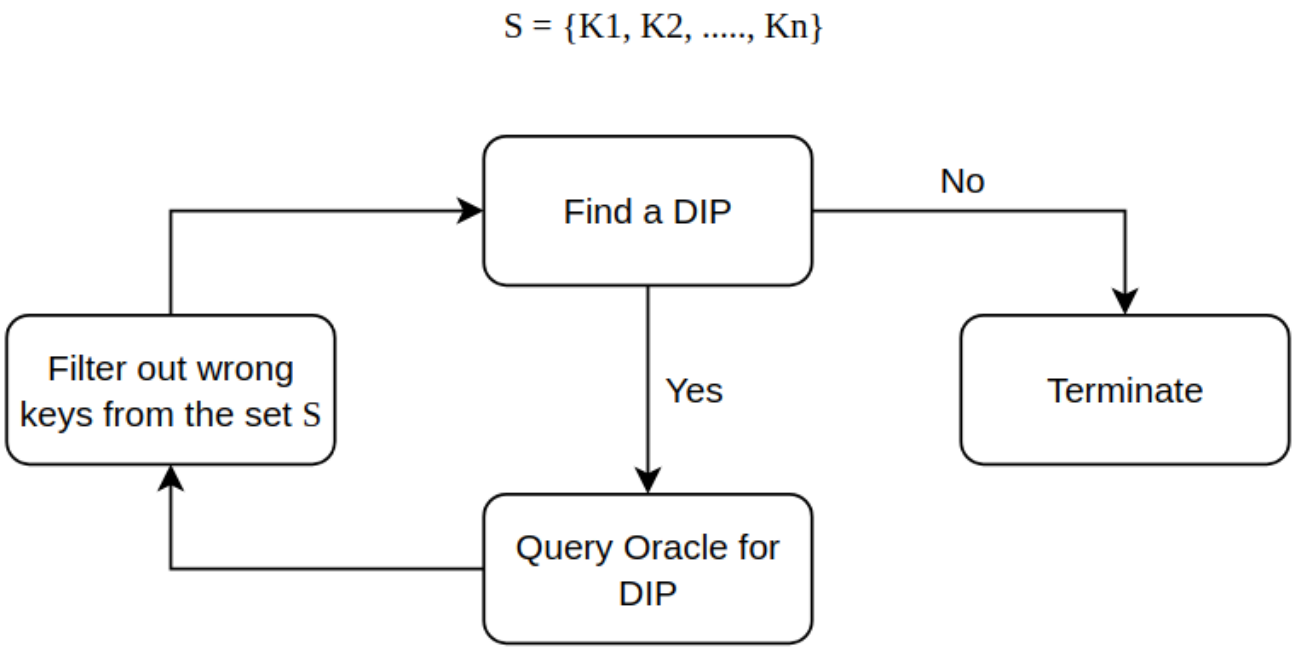


Figure: SMT Attack

Insight

There can be multiple possible parameter sets that can exactly replicate the behavior of the oracle.
Example:

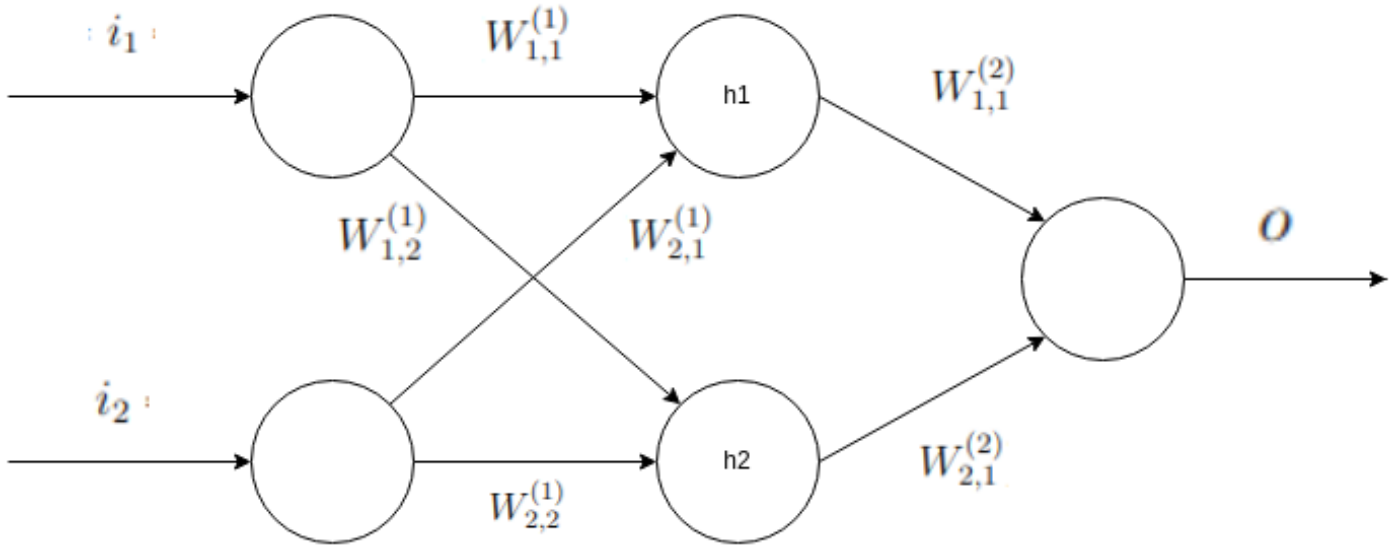


Figure: 2*2*1 NN with no biases

$$o = i_1 * (W^{(1)}_{1,1} * W^{(2)}_{1,1} + W^{(1)}_{1,2} * W^{(2)}_{2,1}) + i_2 * (W^{(1)}_{2,1} * W^{(2)}_{1,1} + W^{(1)}_{2,2} * W^{(2)}_{2,1})$$

Results

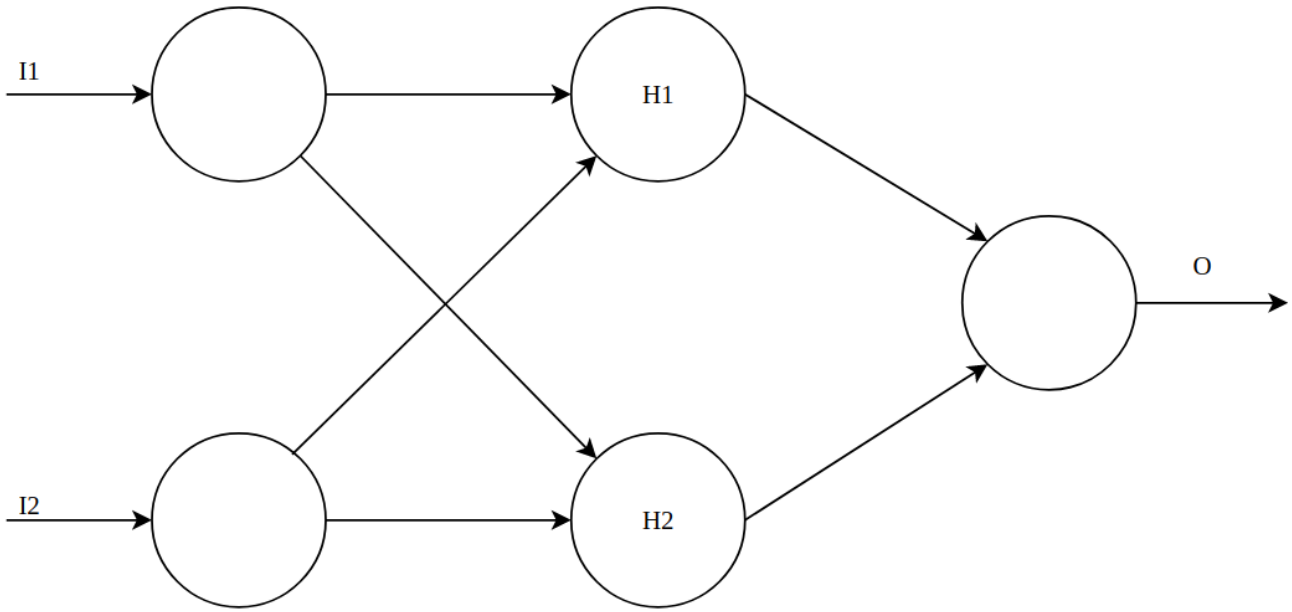
S No	Structure	Bias	Non Linearity	Known Keys	Total Keys	Time Taken	Result
1	2*1	None	None	None	2	0.02 sec	Success
2	4*1	None	None	None	4	0.04 sec	Success
3	1*1*1*1	None	None	None	4	3 sec	Success
4	2*2*1	None	None	None	6	32.97 sec	Success
5	2*2*1	None	2nd layer	None	6	512.77 sec	Success
6	2*2*1	1 at 2nd layer	None	None	7	27.19 sec	Success
7	2*2*1	2 at 2nd layer	None	None	8	96.85 sec	Success
8	2*2*1	1 at 2nd layer	2nd layer	None	7	2460.66 sec	Success
9	2*2*1	2 at 2nd layer	2nd layer	None	8	Infinite	Failed
10	2*2*1	2 at 2nd layer	2nd layer	1 Bias	7	Infinite	Failed
11	2*2*1	2 at 2nd layer	2nd layer	1 weight in L1	7	Infinite	Failed
12	2*2*1	2 at 2nd layer	2nd layer	1 weight in L2	7	Infinite	Failed
13	2*3*1	None	None	None	9	300.72 sec	Success
14	2*3*1	3 at 2nd layer	None	None	12	Infinite	Failed

Table: Data collected from experiments

Observations

1. Time taken increases with an increase in the number of keys.
2. Time taken increases with an increase in non-linearity.
3. Time taken increases with an increase in the number of layers.
4. Time decreases because of the known keys.
5. Impact of non-linearity in the time taken is more than the number of keys.
6. We are able to solve the network with a maximum of 7 keys with non-linearity and 12 keys without non-linearity.

Updated Attack Model



End-End Oracle: {O}, Updated Oracle: {H1, H2, O}

$$h_1 = i_1 * W^{(1)}_{1,1} + i_2 * W^{(1)}_{2,1},\ h_2 = i_1 * W^{(2)}_{1,1} + i_2 * W^{(2)}_{2,1}$$

$$o = i_1 * (W^{(1)}_{1,1} * W^{(2)}_{1,1} + W^{(1)}_{1,2} * W^{(2)}_{2,1}) + i_2 * (W^{(1)}_{2,1} * W^{(2)}_{1,1} + W^{(1)}_{2,2} * W^{(2)}_{2,1})$$

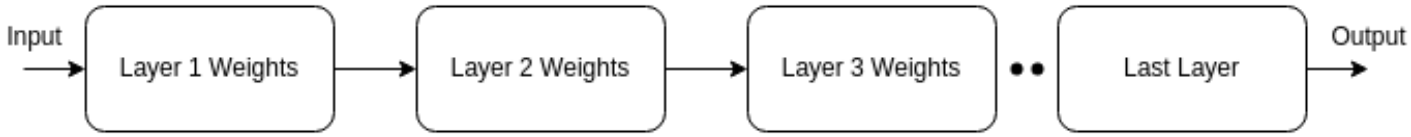
Attack Strategy

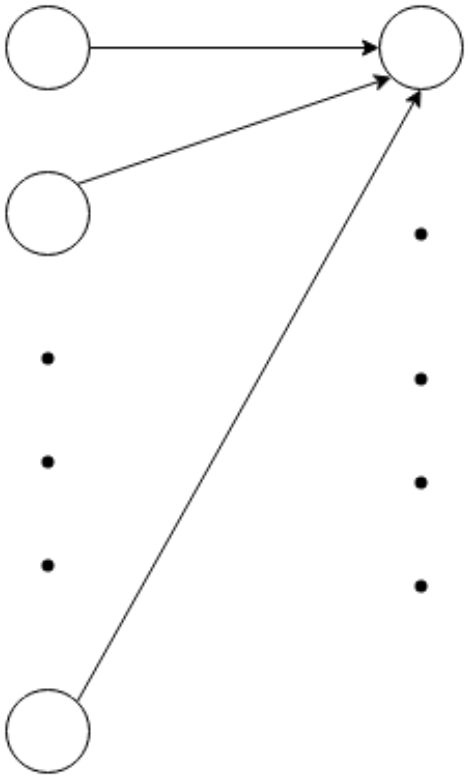
1. We use outputs of all layers to find all parameters at once.
2. Use SMT Attack with all constraints.
3. Now, the solver has more data to work with for the same number of DIP.
4. Reduces the number of queries and time consumption.

Results

S No	Structure	Bias	Non Linearity	Known Keys	Total Keys	Old Time	Updated Time	Queries	Result
1	2*1	None	None	None	2	0.02 sec	0.01 sec	2	Success
2	4*1	None	None	None	4	0.04 sec	0.01 sec	4	Success
3	1*1*1*1	None	None	None	4	3 sec	0.03 sec	2	Success
4	2*2*1	None	None	None	6	32.97 sec	0.04 sec	3	Success
5	2*2*1	None	2nd layer	None	6	512.77 sec	0.25 sec	5	Success
6	2*2*1	1 at 2nd layer	None	None	7	27.19 sec	0.03 sec	3	Success
7	2*2*1	2 at 2nd layer	None	None	8	96.85 sec	2.81 sec	3	Success
8	2*2*1	1 at 2nd layer	2nd layer	None	7	2460.66 sec	0.30 sec	4	Success
9	2*2*1	2 at 2nd layer	2nd layer	None	8	Timeout	0.07 sec	3	Success
10	2*2*1	2 at 2nd layer	2nd layer	1 Bias	7	Timeout	2.74 sec	3	Success
11	2*2*1	2 at 2nd layer	2nd layer	1 weight in L1	7	Timeout	1.22 sec	3	Success
12	2*2*1	2 at 2nd layer	2nd layer	1 weight in L2	7	Timeout	2.31 sec	3	Success
13	2*3*1	None	None	None	9	300.72 sec	0.21 sec	3	Success
14	2*3*1	3 at 2nd layer	None	None	12	Timeout	0.37 sec	3	Success
15	2*3*1	3 at 2nd layer	2nd layer	None	12	Timeout	47.79 sec	3	Success
16	Bool 10*10*10*1	None	None	None	210	Timeout	89.89 sec	162	Success
17	Bool 15*10*10*10*1	None	None	None	360	Timeout	970 sec	288	Success
18	3*4*4*1	None	None	None	32	Timeout	3617 sec	28	Success
19	10*10*10*1	None	None	None	210	Timeout	Timeout	-	Failed

Specific Query Approach





Specific Query Approach

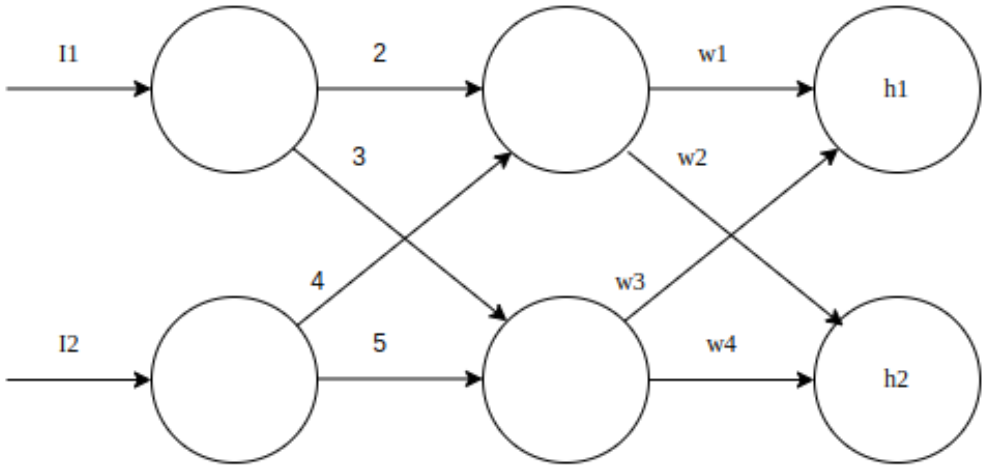
$$h_1 = ReLU(i_1 * W^{(1)}_{1,1} + i_2 * W^{(1)}_{2,1} ... + i_n * W^{(1)}_{n,1})$$

1. To solve an equation of n variables (weights), we need 'n' linearly independent equations.
2. To remove the effect of Non-linearity, we add a constraint output at the node is greater than 0.
3. Keys of our SMT model are inputs of the neural network.
4. We find a DIP such that the linear equation is linearly independent of the previously found DIP's.
5. Corner Case: All weights are 0.

Incremental SMT Attack

1. We go layer by layer, and in each layer we find parameters for each node separately.
2. Keys to our SMT solver are the parameters of the NN.
3. Once, we find the keys of a layer, we fix those values in our structure and start computing for the next layer.
4. The time to break a layer depends on the number of neurons in the previous layer and also the depth of the current layer.

Incremental SMT Attack



$$h_1 = w_1 * (2i_1 + 4i_2) + w_2 * (3i_1 + 5i_2)$$
$$h_2 = w_3 * (2i_1 + 4i_2) + w_4 * (3i_1 + 5i_2)$$

Incremental SMT Attack Results

S No	Structure	Bias	Non Linearity	Total Keys	Previous Time	Incremental Time	Queries	Result
1	3*4*1	None	ReLU	16	80 sec	4 sec	10	Success
2	5*5	None	ReLU	25	Timeout	1.224 sec	40	Success
3	3*4*4*1	None	ReLU	32	3617 sec	70.3 sec	19	Success
4	3*4*4*4	None	ReLU	44	Timeout	196.7 sec	17	Success
5	3*4*4*4	None	ReLU	44	Timeout	636.8 sec	23	Success
6	3*4*4*4*1	None	ReLU	48	Timeout	1528 sec	25	Success
7	3*4*4*4*4*1	None	ReLU	64	Timeout	Timeout	-	Failed
8	5*5*5	None	ReLU	25	Timeout	Timeout	-	Failed

S No	Structure	Non Linearity	Total Keys	Total	Layer 1	Layer 2	Layer 3	Layer 4	Result
1	3*4*1	ReLU	16	4 sec, 10	2.9 sec, 6	1.1 sec, 4	-	-	Success
2	3*4*4*1	ReLU	32	70.3 sec, 19	2.9 sec, 6	66.4 sec, 9	2.27 sec, 4	-	Success
3	3*4*4*4	ReLU	44	196.7 sec, 17	3 sec, 6	72 sec, 9	121.7 sec, 8	-	Success
4	3*4*4*4	ReLU	44	636.8 sec, 23	2.88 sec, 6	69.7 sec, 9	564.26 sec, 8	-	Success
5	3*4*4*4*1	ReLU	48	1528 sec, 25	2.88 sec, 6	69.7 sec, 9	564.26 sec, 8	892 sec, 2	Success
6	5*5*5	ReLU	50	1.224 sec, 40	Timeout	-	-	-	Failed

Table: Layer by Layer data for Incremental SMT Attack

Bounds on Parameters

1. 'Int' datatype used in z3-py has no max limit.
2. In quantized neural networks, the weights are generally 8-bit integers.
3. By keeping bounds on the parameters thereby reducing the key space decreases the overall queries and time consumption of our attack.
4. Two ways to implement are keeping bounds on the keys or replacing 'Int' with 'BitVectors'.

Automation Tool

1. Input: Number of layers, Number of neurons in each layer
2. Input: A file with bounds on each parameter or weight of the neural network.
3. Ouput: Python file with SMT code. (ready to be executed)
4. Can be adapted to different datatypes such as 'Int', 'Real', and 'BitVec' for parameters.
5. Current Assumption: The Oracle is an executable file.

Thank You