APPENDIX A
PROOF OF THEOREM 1

Theorem 1. *For a binary query point x the losses $\mathcal{L}_\phi(x)$ or $V_c(x)$ are equal to 1 if the formula ϕ or clause c is satisfied and 0 otherwise.*

Proof. The idea is to give proof in two steps. First, we prove that when a binary query satisfies the formula, then formula value $\mathcal{L}_\phi(x)$ and all clauses values $V_c(x)$ should be 1. Secondly, when a binary query does not satisfy the formula, at least one clause value and formula value should be 0.

Firstly, assume that the query corresponds to a satisfying assignment, then all clauses are satisfied. That implies that in each clause, there is at least one positive literal with its query value being 1 or a negative literal with its query value being 0. Therefore in each clause loss $V_c(x) = 1 - \prod_{i \in c^+}(1 - x_i) \prod_{i \in c^-} x_i$ at least one of the elements of the product is 0. That implies that all clause losses $V_c(x)$ are equal to 1. Therefore the loss $\mathcal{L}_\phi(x) = \prod_{c \in \phi} V_c(x)$ is 1.

Secondly, assume the query corresponds to an unsatisfying assignment, then there is at least one clause c that is unsatisfied. This implies that in the unsatisfied clause c the query value for all positive literals is 0 and for all negative literals it is 1. Therefore in the clause loss $V_c(x) = 1 - \prod_{i \in c^+}(1 - x_i) \prod_{i \in c^-} x_i$ all elements of the product are 1. This implies that for the unsatisfied clause the corresponding clause loss $V_c(x)$ is 0. Therefore the loss $\mathcal{L}_\phi(x) = \prod_{c \in \phi} V_c(x)$ is 0, since one of the elements of the product is 0. □

APPENDIX B
PROOF OF THEOREM 2

Theorem 2. *A single query that returns the loss V_c for each clause c is sufficient to uniquely identify the SAT formula ϕ to be solved.*

Proof. The idea is to choose the query input values based on primes such that the clause loss contains an irreducible fraction with the numerator and denominator consisting of prime factors which point to the variables that appear in the clause. From the measured clause loss value, it is possible to infer the corresponding irreducible fraction, and from this fraction, we can infer the variables and the signs of their corresponding literals in the clause.

An element of the query $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ corresponds to a value of a variable queried at an intermediate (real) point between 0 and 1, where n is the number of variables. Each query element is obtained by using a pair from the series of prime pairs where the gap between the two primes in the pair is a fixed constant H. For $H = 2$ the series would be pairs of twin primes: $(3, 5), (5, 7), (11, 13), \ldots$. It has been proven that there is a constant H between 2 and 246 such that the corresponding prime pair series contains an infinite number of elements [27]. We fix a concrete value of H for constructing the query.

For constructing the query from the series, we take the first n pairs $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$ that fulfill the following criteria: $\forall_i\ a_i > H$ and $\forall_{i,j}\ a_i \neq b_j$. Note that $b_i - H = a_i$. We set the query x as $(H/b_1, H/b_2, ..., H/b_n)$.

Let us examine the clause loss $V_c(x) = 1 - \prod_{i \in c^+}(1 - x_i) \prod_{i \in c^-} x_i$ corresponding to a clause c and the query x with i positive and $k - i$ negative literals. To simplify the notation, we assume without a loss of generality that the first k variables appear in the clause and that the positive literals have lower indices than negative literals.

$$
\begin{aligned}
V_c(x) &= 1 - (1 - x_1)(1 - x_2) \ldots (1 - x_i) x_{i+1} x_{i+2} \ldots x_k = \\
&= 1 - \left(1 - \frac{H}{b_1}\right)\left(1 - \frac{H}{b_2}\right) \ldots \left(1 - \frac{H}{b_i}\right) \frac{H}{b_{i+1}} \frac{H}{b_{i+2}} \cdots \frac{H}{b_k} = \\
&= 1 - \frac{b_1 - H}{b_1} \frac{b_2 - H}{b_2} \cdots \frac{b_i - H}{b_i} \frac{H}{b_{i+1}} \frac{H}{b_{i+2}} \cdots \frac{H}{b_k} = \\
&= 1 - \frac{a_1}{b_1} \frac{a_2}{b_2} \cdots \frac{a_i}{b_i} \frac{H}{b_{i+1}} \frac{H}{b_{i+2}} \cdots \frac{H}{b_k} = 1 - \frac{a_1 a_2 \ldots a_i H^{k-i}}{b_1 b_2 \ldots b_k}
\end{aligned}
$$

The indices of the primes $b_1 b_2 \ldots b_k$ in the denominator of the fraction in the obtained expression correspond to the indices of variables appearing in the clause c. The indices of the primes $a_1 a_2 ... a_i$ in the numerator correspond to the indices of variables appearing as positive literals in the clause.

The primes $a_1 a_2 \ldots a_i$ and $b_1 b_2 \ldots b_k$ are non-overlapping and the prime factors of H are smaller than any prime a_i or b_i due to the pair selection criteria. This implies that the fraction in the representation of the clause loss that we obtained is irreducible.

We prove that given the constructed query x, the clause loss function is injective. To show that, we take a clause c' that is different from our arbitrarily chosen clause c. The clause c' being different from clause c implies that it will differ in at least one literal, and therefore the irreducible fraction in the corresponding clause losses $V'_c(x)$ and $V_c(x)$ will differ in $a_1 a_2 \ldots a_i$ or $b_1 b_2 \ldots b_k$. Due to the fundamental theorem of arithmetic, there is only one way to write a number as a product of its prime factors (up to the order of the factors). The numerator or the denominator of the fraction in the losses will differ because their prime factors will differ. Therefore any two different clauses will produce clause losses that each contain a different irreducible fraction. Since each rational number can be written as an irreducible fraction in exactly one way, the two clause losses are different rational numbers, and hence the clause loss function is injective for the constructed query.

Therefore the clause can be identified from a query and its corresponding loss for the clause. Since the instance ϕ to be solved is uniquely represented by its constituent clauses, it can be identified from a query and a set containing a clause loss for each of its clauses. □

APPENDIX C
QUERY RESULTS

To analyze how the model uses queries and what information they contain, we train NeuroCore with the query mechanism on the 3-SAT task for 100k iterations as described in the section V. At each recurrent step, query and logit values are discretized by applying the sigmoid function and then rounding values to the closest integer (0 or 1). Fig. 10 reveals how many clauses the

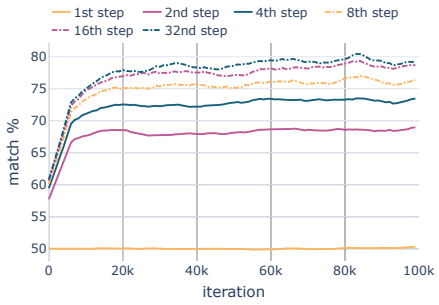

Fig. 9: Match between query and logits (higher value means more similar) of the same recurrent step depending on the training iteration on the 3-SAT task.

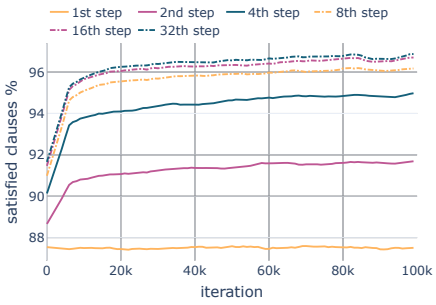

Fig. 10: Per cent of clauses satisfied by query depending on the training iteration on the 3-SAT task.

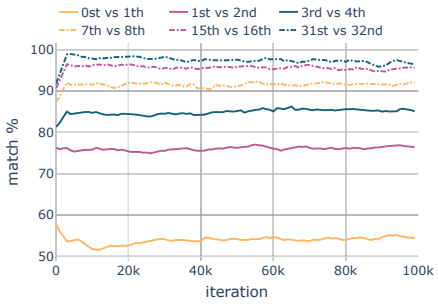

Fig. 11: Element-wise match (higher values indicate greater similarity) between two queries in consecutive steps depending on the training iteration on the 3-SAT task.

discretized query satisfies at each step in the training time. Fig. 9, on the other hand, depicts an element-wise match between discretized queries and logits as per cent of identical variables from the total variable count. In Fig. 9, we can see that the queries have different values from the logits of the same step and from the Fig. 10 we can see that they never satisfy given Boolean formula, strengthening our belief that queries are not used to output the true prediction but instead to obtain rich information about the problem.

We also validate that model produces different queries at each recurrent step, therefore obtaining rich information about the problem. Fig. 11 depicts the match as per cent of equal elements from the total element count between two discretized queries of two consecutive steps. As reasoned in section III-B, query mechanism can extract more information about the problem by issuing several different queries at each step and accumulating this knowledge. In Fig. 11, we can see the difference between the last and second last query increase with the training time. That indicates that the model learns to issue more different queries, increasing the obtained information about the problem.