Superiority of the SOL Algorithm in Learning Logical Functions

Keisuke Shindo

April 4, 2025

Abstract

This paper presents a novel machine learning algorithm called SOL. We demonstrate that SOL significantly outperforms conventional models such as Multi-Layer Perceptrons (MLPs) and Transformers in terms of accuracy and learning efficiency when learning logical functions. Experimental results show that SOL achieves perfect accuracy (1.0) with fewer learning steps, whereas MLP and Transformer models fail to reach complete accuracy. These findings suggest that SOL is a promising approach for tasks requiring logical reasoning.

1 Introduction

Multi-Layer Perceptrons (MLPs) and Transformers are widely used models in machine learning, especially for classification and regression tasks. However, these models have limitations in precisely learning logical functions. To address this issue, we propose a new algorithm named SOL, specifically designed for high-accuracy learning of logical functions. This study compares SOL with MLPs and Transformers in terms of accuracy, training time, and convergence speed.

2 Method

We implemented the following three models:

- **SOL Algorithm**: A new approach specialized in learning logical functions.
- **MLP Model**: A conventional neural network with 128 neurons in the hidden layer.
- **Transformer Model**: A standard Transformer-based model with an attention mechanism.

Each model was trained on a dataset of 1024 samples (X, Y), where $Y[16] = f(X[32])$, constructed from the logical expressions below. The models were evaluated based on bitwise accuracy, overall accuracy, and learning efficiency.

Table 1: Training logics for the models

$$
\begin{aligned}
Y[0] &= X[0] \land X[1] \\
Y[1] &= X[2] \lor X[3] \\
Y[2] &= X[1] \oplus X[3] \\
Y[3] &= \neg X[0] \land X[1] \\
Y[4] &= \neg (X[0] \land X[1]) \\
Y[5] &= \neg X[2] \lor X[3] \\
Y[6] &= \neg X[0] \oplus X[1] \\
Y[7] &= \neg (X[2] \land X[3]) \\
Y[8] &= X[0] \land X[1] \land X[2] \land X[3] \\
Y[9] &= X[0] \lor X[1] \lor X[2] \lor X[3] \\
Y[10] &= X[0] \oplus X[1] \oplus \neg X[2] \oplus X[3] \\
Y[11] &= (X[0] \lor X[1]) \land (X[2] \lor X[3]) \\
Y[12] &= X[0] \lor \neg X[1] \lor X[2] \lor X[3] \\
Y[13] &= (X[0] \oplus X[1]) \lor (X[2] \oplus X[3]) \\
Y[14] &= (X[0] \oplus \neg X[1]) \land (X[2] \oplus \neg X[3]) \\
Y[15] &= (X[0] \lor X[1]) \oplus (\neg X[2] \land X[3])
\end{aligned}
$$

The input data X consists of 32-bit vectors with varying bits. The SOL algorithm was trained only once using the 1024-sample dataset, while MLP and Transformer models underwent multiple epochs of training using the same dataset.

3 Experimental Results

Table 2 presents the accuracy and training time for each model.

Table 2: Accuracy and training time for each model

Model	**Accuracy**	**Training Steps**	**Training Time (s)**
SOL	1.0000	1024 steps	2811.72
MLP	0.9656	3500 epochs	3523.27
Transformer	0.7211	400 epochs	4097.79

Table 3 shows the bitwise accuracy for each model. SOL achieved perfect accuracy across all logical functions. In contrast, MLP and Transformer models tended to perform poorly on XOR-based functions such as $Y[2]$, $Y[10]$, and $Y[15]$.

Table 3: Bitwise accuracy per output bit $Y[0]$ to $Y[15]$

Model	**Bitwise Accuracy**
SOL	1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000
MLP	1.0000, 1.0000, 0.7500, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 0.7330, 1.0000, 1.0000, 1.0000, 1.0000, 0.8370
Transformer	0.7860, 0.7270, 0.5260, 0.7710, 0.7860, 0.7040, 0.5200, 0.7690, 0.9530 ,0.9250, 0.5090, 0.5040, 0.9500, 0.7410, 0.7410, 0.6260

4 Test environment

- SOL implementation
`https://github.com/KeisukeShindo0/SOL_feedback_evaluation/tree/main/src/SOLImplementation`
`py`
- MLP Model
`https://github.com/KeisukeShindo0/SOL_feedback_evaluation/tree/main/src/SOL_compare_MLP.py`
- Transformer model
`https://github.com/KeisukeShindo0/SOL_feedback_evaluation/tree/main/src/SOL_compare_Transformer.py`

5 Discussion

The experimental results show that the SOL algorithm significantly outperforms both MLP and Transformer models in learning logical functions. While the MLP achieved relatively high accuracy, it required a longer training time. The Transformer struggled with certain logical functions, indicating that it may not be well-suited for this type of task. Moreover, SOL achieved high accuracy with fewer training steps, demonstrating superior computational efficiency.

6 Conclusion

We proposed a new machine learning algorithm, SOL, capable of learning logical functions with high accuracy. Our experiments confirmed that SOL is superior to conventional MLP and Transformer models in terms of both accuracy and efficiency. Future work will explore extensions to more complex logical structures and real-world applications.