Review Article

A Review on Designing of Future State Estimation of Finite State Machine By Deep Learning Algorithm

Akash Kaushal[1], Prof. Nishi Pandey[2], Prof. Abhishek Agwekar[3]

[1]*M.Tech. Scholar, Truba Institute of Engineering & Information Technology, Bhopal, (M.P.), INDIA*
[2]*Assistant Professor, Truba Institute of Engineering & Information Technology, Bhopal, (M.P.), INDIA*
[3]*HOD, EC, Truba Institute of Engineering & Information Technology, Bhopal, (M.P.), INDIA*

ABSTRACT

The aim is to design an Artificial Intelligence based system which would get trained using the limited input-output mapping of the circuit and be able to predict the future behaviour of the circuit. This would make digital circuits interactive. In the present work, Artificial Neural Networks have been used for predicting the behaviour of digital circuits. In the present work, different digital circuits have been simulated such as the AND Gate, Half Adder, JK Flip Flop and Sequence Detecting Finite State Machines using AI based systems. As a standard convention, 70% of the total data has been employed for training, and the rest of the 30% has been employed for testing and validation. It has been found that the behaviour of these circuits can be predicted with ease using Artificial Neural Networks. Predictive modelling for finite state machines has been done by using a sequence detector as a finite state machine. The number of inputs of the finite state machine is varied from 8 to 64. The variation is brought about in terms of 8 input bits or 1 byte. Finally, the Error Histogram analysis for Finite State machines with different inputs has been performed and presented. A comparative analysis with previous work has also been presented. It can be clearly seen that the proposed system outperforms the previous work in terms of the performance parameters of the designed system.

KEYWORDS

Artificial Intelligence, Artificial Neural Networks, Finite State Machine, Levenberg Marqardt Algorithm, Mean Square Error

I. INTRODUCTION

1.1 FINITE STATE MACHINES

State machines are nothing but a method of modelling a sequential circuit whose output is depend on both the current and the previous inputs. If we compare a state machine with a purely functional system whose output is a function of input then state machine uses the history of inputs for the operation. There are a lot of systems which are designed using a state machine; some of them are given below.

- User interfaces, with typed input, mouse clicks, etc.;
- Conversations, in which, for example, the meaning of a word "it" depends on the history of things that have been said;
- The state of a spacecraft, including which valves are open and closed, the levels of fuel and oxygen.
- The sequential patterns in DNA and what they mean.

State machine models can either be continuous time or discrete time. In continuous time models, we typically assume continuous spaces for the range of values of inputs and outputs, and use differential equations to describe the system's dynamics. In discrete-time models inputs and outputs are determined at specific increments of time, and which are synchronized to those specific time samples.

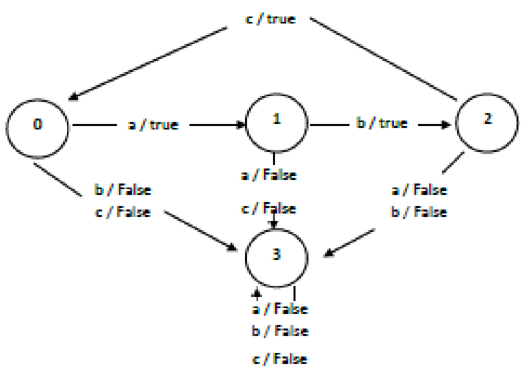

Figure 1.1: State transition diagram

1.2 Mealy Machine

In this model of FSM, the output values are determined both by its current state and the current inputs. In the figure 1.2 it is shown that the input is fed to the combinational circuit C1 and then a state register is used to store the output of C1. The input and the output of the register fed to the second combinational circuit C2. Hence the output is determined both by present state and input.

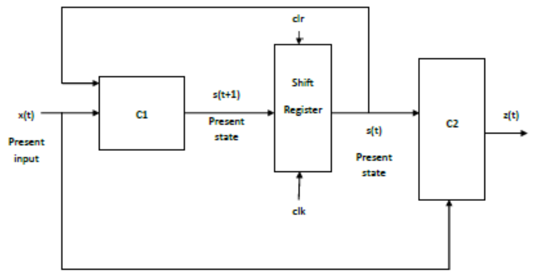

Figure 1.2: Mealy Machine

1.3 Moore Machine

In this model of FSM, the output values are determined by its current state only. In the figure 1.3 it is shown that the input is fed to the combinational circuit C1 and then a state register is used to store the output of C1. The output of the register fed to the second combinational circuit C2. Hence the output is determined by present state only.

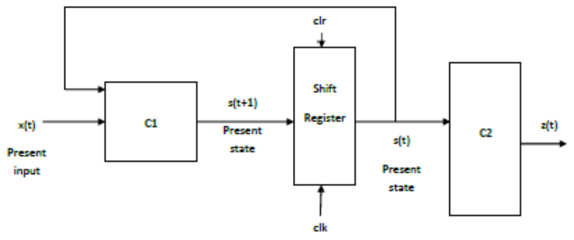

Figure 1.5: Moore Machine

1.4 Artificial Intelligence

Artificial intelligence can be defined as the development of computational systems competent enough to perform tasks which would otherwise require human intervention.

The fundamental requirement of understanding artificial intelligence is the perception of intelligence in the first place. Intelligence can be thought of as a sequential implementation of the following steps:

1) Accepting data.

2) Analysing Data.

3) Figuring out regularities or patterns in the data.

4) Taking some decision based on the above steps.

5) Storing the experiences from every input and decision taken thereafter.

6) Making use of the stored experiences in taking future decisions

One of the most common techniques of developing and implementing a system with artificial intelligence is the design of Artificial Neural Networks (ANN). The human brain is made of brain cells called neurons. Therefore the human brain is called a neural network. An artificial neural network is a computational machine which emulates the human neural network (brain).

II. LITERATURE SURVEY

Song et.al. in the paper “Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control” proposed an approach to investigate the spiking neural P systems, i.e. SN P systems, with astrocyte-like control which were proven to have “Turing completeness” as computing models. In biological nervous systems, the operation of interacting neurons depends largely on the regulation from astrocytes. In this research, it is proposed and established SN P systems with astrocyte-like control to emulate logic AND, OR, NOT, NOR, XOR and NAND gates. The obtained SN P systems are simple and homogeneous, which infers that each neuron in the systems has the same unique spiking rule. The systems proposed in this work provide theoretical models to construct neural-like logic gates with universal information processing units. It will be beneficial if parallel hardware, such as GPU, can be utilized and harnessed to realize the neural-like digital logic gates and logic circuits.

Eisner el al. in the paper “Weighting Finite-State Transductions With Neural Context” proposed a new approach to deep learning to tasks such as morphological re inflection, which stochastically edits one string to get another .The traditional architecture is kept as such, and A stack of bidirectional LSTMs reads the input string from left-to-right and right-to-left, in order to summarize the input context in which a transducer arc is applied. These learned characteristics and traits are intertwined with the transducer to define a probability distribution over aligned output strings, in the form of a weighted finite-state automaton. This lessens hand-engineering of features, allows learned features to examine unbounded context in the input string, and still permits exact inference through dynamic programming.

Kayri et al. in the paper “Predictive Abilities of Bayesian Regularization and Levenberg– Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data “demonstrated the objective of the study which was to compare the predictive ability of Bayesian regularization with Levenberg–Marquardt Artificial Neural Networks. Since complex models are penalized in accordance with the Bayesian approach, this approach explores complex architecture smoothly. All in all, the model with two-neurons is the best architecture because of the highest importance level of predictors.

Gross et al. in the paper “VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing” suggested that an integer form of stochastic computation and introduce some elementary circuits. An efficient implementation of a DNN based on integral stochastic computing is put forward. An efficient stochastic implementation of a deep belief network is proposed using integral SC. Going by the simulation and implementation results ,it show that the proposed design reduces the area occupation by 66% and the latency by 84% with respect to the state of the art. It also indicates that the proposed design consumes 21% less energy than its binary radix counterpart. Moreover, the proposed architectures can save up to 33% energy consumption w.r.t. the binary radix implementation by using quasi-synchronous implementation without compromising on performance and keeping the efficiency intact.

Abubakar et al. in the paper "State assignment for area minimization of sequential circuits based on cuckoo search optimization "proposed the application of cuckoo search optimization (CSO) algorithm for solving the state assignment problem (SAP) of FSMs with the aim of minimizing area of the resulting sequential circuit. Results obtained from the CSO algorithm are compared with those obtained from binary particle swarm optimization (BPSO) algorithm, genetic algorithm (GA), and the well-known deterministic methods of NOVA and JEDI. The results indicate that CSO outperforms deterministic methods as well as other non-deterministic heuristic optimization methods. It presented the application of a recent optimization method, the cuckoo search optimization algorithm, to solve the state assignment problem in FSM.

Ławry´nczuk et al. in the paper "Jordan Neural Network for Modelling and Predictive Control of Dynamic Systems" proposed and discussed the possibility of using a Jordan neural network as a model of dynamic systems and it highlights a Model Predictive Control (MPC) algorithm in which such a network is used for prediction. The Jordan network is a simple recurrent neural structure in which only one value of the process input signal (from the previous sampling instant) and only one value of the delayed output signal of the model (from the previous sampling instant) are used as the inputs of the network. It can be successfully used in predictive control.

Cavalli et al. in the paper "On adaptive experiments for nondeterministic finite state machines" proposed an algorithm for deriving adaptive homing and distinguishing experiments for non initialized nondeterministic finite state machines (NFSMs). Given a non-initialized complete FSM, a way for deriving adaptive homing/distinguishing experiments is proposed. Adaptive experiments are represented as special nondeterministic observable machines, called test cases. Necessary and sufficient conditions for having adaptive homing/ distinguishing test cases with minimal length for observable and non-observable nondeterministic FSMs are established.

Schmidhuber et al. in the paper "Deep Learning in Neural Networks: An Overview" proposed a study on Deep Learning in Neural Networks. Standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment and other neurons get activated through weighted connections from previously active neurons. By triggering actions some neurons may affect the environment. Learning or credit assignment is about finding weights that make the NN exhibit desired behaviour, such as driving a car. Depending on the specific problem and cognizance of how the neurons are connected, such behaviour may require long causal chains of computational stages, where each stage transforms (often in a non-linear way) the aggregate activation of the network. Deep Learning is about accurately assigning credit across many such stages.

Drechsler et al. in the paper "A Synthesis Flow for Sequential Reversible Circuits" proposed a paper on a synthesis flow for sequential reversible circuits. It is a powerful methodology that was introduced which transformed a finite state machine into a Boolean function representing the sequential behaviour. Heuristics ensure that encodings for the states are applied due to which the costs of the resulting circuits remain low.

Margaretha et al. in the paper "Back propagation and Levenberg-Marquardt Algorithm for Training Finite Element Neural Network" carried out a study on development of finite element based neural network. Finite element method is combined with artificial neural network using back propagation algorithm to solve differential equation and Levenberg-Marquardt training algorithm to solve inverse differential problem. By use of this devised method, inverse matrix calculation will not be necessary for solving both differential equation and inverse differential problem making it simple to implement. Experimental results also show good accuracy for solving some problems. Future research includes extending this method for 3-D and non-linear finite element. Training algorithm for neural network can be modified using conjugate gradient method, or using neural network architecture, like Hopfield network. Combining this method with genetic algorithm also noted to be interesting.

Skliarova et al. in the paper, "Design and Implementation of Parallel Hierarchical Finite State Machines", proposed a now model and method for synthesis of parallel hierarchical finite state machines (PHFSM) which made it possible to implement algorithms comprising of small modules such that the one module could be activated by another module and multiple modules could be activated in parallel. This technique thus made it possible to implement the cascading concept in finite state machines. It also paved the path for the parallel design of finite state machines. The designs were in actuality implemented and tested on FPGAs and their efficacy was proven in solving real life problems.

III. PROBLEM DOMAIN

The design of such systems requires the need of Artificial Intelligence. There are several AI based algorithms that can be utilized for the design and prediction of digital circuits. We mainly focus on systems with feedback or systems with **Back Propagation.**

The evaluation of the performance of the designed AI based system generally is computed on the basis of the following performance metrics:

1) Mean Square Error (MSE): It gives an idea about the difference in predicted output and actual output (ideal output or target)

IV. PROPOSED METHODOLOGY

Artificial Neural Networks (ANN) are computing systems or technique that are inspired by the learning architecture of human brain to discover the relations between the input and target variables of a system. Human brain consists of a large set of structural constituents, known as neurons, which form a well-connected network to respond to an input signal to perform all its computations / calculations in a certain complex task such as image and voice recognition task and they do this with incredible speed and accuracy. Neurons are simple processing units, which has the ability to store experimental data and which work as parallel distributed processor. The speed of human brain is several thousand time faster than traditional computer because in brain unlike traditional computer as whole information is not passed from neuron to neuron they are rather encoded in the neuron network. This is reason why neural network is also named as connectionism.

A biological model of neuron is basically comprised of dendrites, a cell body or soma, and an axon. The cell body, also called the soma, holds the nucleus of neurons. The dendrites are the branches that are linked to the cell body and stretched in space around the cell body to receive signals from neighbouring neurons. The axon works as a transmitter of the neuron. It sends signals to neighbouring neurons. The synapse or synaptic terminal are the connection between the axon of one neuron and the dendrites of neighbouring neutron, which is the communication link in between the two neurons.

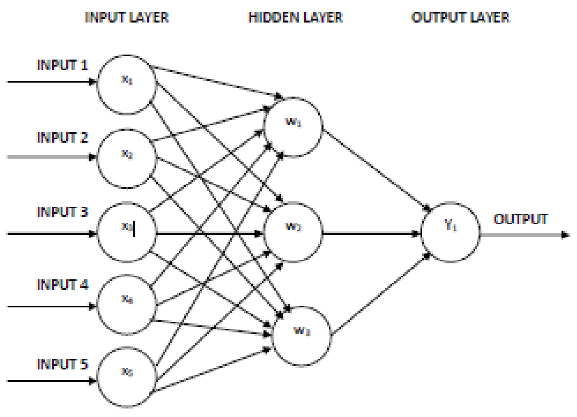

Figure 4.1: Working model of an ANN

V. RESULTS AND DISCUSSIONS

Before proceeding towards complex FSMs, we start our discussion with the design and implementation of logic gates. Subsequently, we move on to the design of combinational circuits, sequential circuits and finally sequence detectors.

In all the above examples, the customary rule of taking 70% of the data for training and remaining 30% of the data for testing and validation has been followed. The circuits simulated are:

1) AND GATE using LM Algorithm

2) HALF ADDER using LM Algorithm

3) JK FLIP FLOP using LM Algorithm

4) A Jordan network implementation of the Half Adder

5) Sequence detector design using LM, BR and SCG algorithms. The performance indices considered here are:

1) Mean Square Error (MSE)

2) Epochs or execution time

3) Iterations

4) Error Histograms for Graphical Error Representation

5) Regression

VI. CONCLUSION AND FUTURE SCOPE

CONCLUSION

It can be concluded from the previous discussions that the design and prediction of complex digital circuits is possible using Artificial Neural Networks (ANN). The benefit of such a system lies in the fact that there is no need of knowing the internal circuitry or the complete input output mapping of the digital system to predict its performance. The behavior of the circuit can be predicted by the ANN based system once the system has been trained. This approach finds extensive application in interactive gaming, Human Machine Interfaces (HMI), hardware level cryptography. In the present work, different digital circuits have been simulated such as the AND Gate, Half Adder, JK Flip Flop and Sequence Detector using AI based systems. As a standard convention, 70% of the total data has been employed for training, and the rest of the 30% has been employed for testing and validation. It has been found that the behavior of these circuits can be predicted with ease using Artificial Neural Networks. A comparative error analysis clearly shows that the proposed system outperforms the previous system. It can be seen that as the number of inputs increases, the errors also increase due to the complexity of the system. The proposed system attains a maximum error of 0.087 is is relatively 10 time lesser than the error performance of the base paper [1].

FUTURE SCOPE

The future scope can be thought of as in improving and optimizing the present algorithms to attain even better performance indices such as lesser number of iterations, better regression and lesser MSE. Also, the design of cascaded algorithmic structures can be designed to while retaining the merits of each of the algorithms. Also, more complex circuits can be tested in real time applications.

REFERENCES

[1] Ji Li1, Ao Ren2, Zhe Li, Caiwen Ding, Bo Yuan, Qinru Qiu2 and Yanzhi Wang "Finite State Machine SoC Design Using Deep Neural Networks for Future State Estimation," IEEE 2022.

[2] Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control Tao Song, Pan Zhen, M.L. Dennis Wong, Xun Wang, Elsevier 2021

[3] Weighting Finite-State Transduction swith Neural Context Pushpendre Rastogi and Ryan Cotterell and Jason Eisner, NAACL-HLT Proceedings 2020

[4] Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data Murat Kayri, MDPI, 2019

[5] VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing ArashArdakani, Student Member, IEEE, François Leduc-Primeau, NaoyaOnizawa, Member,

IEEE, Takahiro Hanyu, Senior Member, IEEE and Warren J. Gross, Senior Member, IEEE 2018

[6] State assignment for area minimization of sequential circuits based on cuckoo search optimization q Aiman H. El-Maleh, Sadiq M. Sait, Abubakar Bala Elsevier 2017

[7] Jordan Neural Network for Modelling and Predictive Control of Dynamic Systems Antoni Wysocki and MaciejŁawry´nczuk, IEEE 2016

[8] On adaptive experiments for nondeterministic finite state machines Natalia Kushik · Khaled El-Fakih · Nina Yevtushenko · Ana R. Cavalli, Springer 2015

[9] Deep Learning in Neural Networks: An Overview Technical Report IDSIA-03-14 / arXiv:1404.7828 v4 [cs.NE] (88 pages, 888 references) J¨urgen Schmidhuber the Swiss AI Lab IDSIA IstitutoDalleMolle di Studisull'Intelligenza Artificial, IEEE 2014

[10] A Synthesis Flow for Sequential Reversible Circuits Mathias Soeken Robert Wille Christian Otterstedt Rolf Drechsler, IEEE 2014

[11] Back propagation and Levenberg-Marquardt Algorithm for Training Finite Element Neural Network Arnold Reynaldi, Samuel Lukas, Helena Margaretha, IEEE 2012.

[12] RH Byrd, GM Chin, W Neveitt, J Nocedal, "On the use of stochastic hessian information in optimization methods for machine learning", SIAM Journal of Optimization 2011

[13] J Rhinelander, XP Liu, "Stochastic subset selection for learning with kernel machines", IEEE 2011.

[14] J Bae, LS Giraldo, P Chhatbar, J Francis, "Stochastic kernel temporal difference for reinforcement learning", IEEE 2011

[15] F Orabona, L Jie, B Caputo, "Online-batch strongly convex multi kernel learning", IEEE 2010

[16] L Bottou, "Large-scale machine learning with stochastic gradient descent", Springer 2010.

[17] S Yi, D Wierstra, T Schaul, J Schmidhuber, "Stochastic search using the natural gradient", ACM 2009

[18] P Vamplew, R Dazeley, E Barker, A Kelarev, "Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks", Springer 2009

[19] J Ye, JH Chow, J Chen, Z Zheng, "Stochastic gradient boosted distributed decision trees", ACM 2009

[20] Design and Implementation of Parallel Hierarchical Finite State Machines Valery Sklyarov, IouliiaSkliarova, IEEE Explore 2008