International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

Sequential Estimation of Feed Forward Networks for Small Embedded Hardware

Rajarajan G, Madhur Bhatnagar, Sharoni Roy Chowdhury

Abstract*: – *Deterministic Finite Automata (DFA) and Non-Deterministic Automata (NFA) Systems require an input and lead to an output after stage traversals and based on the pathway so chosen leads to the either the acceptance or rejection of a language. Considering compilers, the compiler works first to understand the lexical correctness of the input and to do so follows steps to check for the validity of the same. If the input is of a valid form then the input is accepted else a suitable corresponding error is thrown. When considering a feed forward neural network, we see a pattern of an input being taken and passed to a hidden layer, which further may either pass to another hidden layer (making it a deep network) or lead it to an output layer. Neural networks find application in classification problems, regression analysis and recognition paradigms. On naïve speculation, a correlation can be made on similarities between finite automata and feed forward networks.

Keywords: Automata, Deterministic, Feed forward, Finite, Neural Networks, Non-Deterministic.

I. INTRODUCTION

To estimate the parallel architecture to a sequential one we must try to find a correlation between parallel and reduce it back to its equivalent sequential architecture [2]. Since, A simple parallel architecture requires sequential to parallel distribution, However, the reverse process is not an easy approach since we do not have fixed boundaries for their re-alignment [1][6]. Neural networks are different as they involve a dynamic learning process for their parallelization and hence cannot be statically mapped with fixed routines. Weight adjustments and parallel constructs are key for neural networks which one sequential analysis may not produce the same result or with same efficiency [3][4]. The primary trade off would be the loss of efficiency as parallelization would lead to faster implementation but since we are concerned with small-medium embedded devices we would focus on memory constraints [5], since neural network architectures are heavily dependent on memory as well[9].

Manuscript published on November 30, 2019.
* Correspondence Author
Rajarajan G*, Assistant Professor, School of Computer Science and Engineering, Vellore Institite of Technology, Vellore, India. Email: rajarajan.g@vit.ac.in
Madhur Bhatnagar, School of Computer Science and Engineering, Vellore Institite of Technology, Vellore, India. Email: madhur.bhatnagar2016@vitstudent.ac.in
Sharoni Roy Chowdhury, , School of Computer Science and Engineering, Vellore Institite of Technology, Vellore, India. Email: sharoni.roychowdhury2016@vitstudent.ac.in

II. MODEL ANALYSIS

In order to estimate a possible backward analysis for the current scenario, we need to imagine the Neural as one of the possible estimations from a sequential mapping. What this means is that we need to begin with the assumption that there exists a sequential formation is one of the many possible scenarios, and that sequential estimation is equivalent in nature to the output that is obtained from the feed forward neural network. After this assumption, we analyze the neural network as a deterministic finite machine. This DFA is essentially that sequential path that leads to our output that is equivalent to the neural network we expect. The input layer if considered from parallel to sequential now represents the same input pattern that is followed by a DFA while still retaining the similar structure of a visible layer that is a part of the architecture diagram as shown in Fig. 1. Now comes the estimation of the neural network architecture. Since an input is passed thought different activation functions, it is not difficult to see that by sequentially aligning and keeping track of the previous functions, we can aim to mimic the overall alteration that happens to the input during the hidden layer and essentially "throw" the cumulative alteration as output. Designs must be kept however to retain information of the hidden functions, which can be estimated via backtracking, weight readjustments and repeated epochs [2].

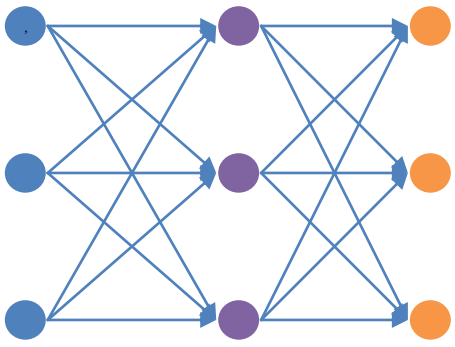

Fig. 1. Standard Representation of a feed forward Neural Network.

III. MAPPING PROCESS

A Deterministic Finite Automata begins with a start state and for every possible input that can be taken from its set of input states, decides the next state to proceed with. Analogous to that is the Feed forward Network.

Let us consider a network that has a visible layer, a hidden layer and an output layer. The input layers are nodes that take the input and forward that input on modification with a weight and passed to the hidden layer. The job of the hidden layer is to take the many inputs and jointly check with some activation function [8][11].

The activation function then decides the output which is forwarded to the output layer. The following diagram in Fig. 2 is an example of a feed forward network. The blue, yellow and green layers are input, hidden and output layers respectively. The Fig. 1. can now be simplified to a representation where the blue node is the start state, the yellow node is an intermediate state and the green node with concentric rings is the final state. The above diagram is a representation of the Feedforward network; however it is not a DFA as we have not defined the DFA with all required parameters.

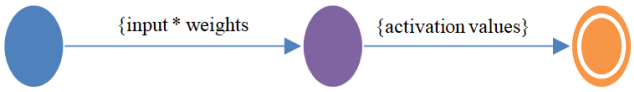

Fig. 2. Simplified understanding of the standard Feed forward Network.

Let now consider the following, given parallel nodes in the input layer of the network, the DFA can resemble a single point of input for the same. We know that a in DFA a given input must follow a deterministic path and every time the same input is given; the same path should be followed. To simplify the explanation, let us look at all inputs as a vector. Hence, for the given example, we can have a X(input) = [x1 x2 x3] that goes to the hidden layer F = [f1 f2 f3] where each f is a function (like tanh or sigmoid etc.) and finally we have the output layer Y = [y1 y2 y3] which holds the outputs of the network. Weights W = [w1 w2 w3] must also be mentioned as they decide how much relevance an input has on the output. Considering the pathway for an input x1, we see that x1 is provided to all hidden layers. Hence x1 becomes $x_{1*}w_i$ where i is {1,2,3}. Then, we see that Y = F(X*W) such that all f_i works with X*W. and based on the output is given to y_i for all the F outputs. We can clearly represent the input and weights as real numbers. So we define an input tuple (x,w) for the DFA where x, y ∈ R. Every tuple (x,w) is passed to every function which again in turn provides a function output which is added to give an output value y_i. We get the following representation as shown in Fig. 3.

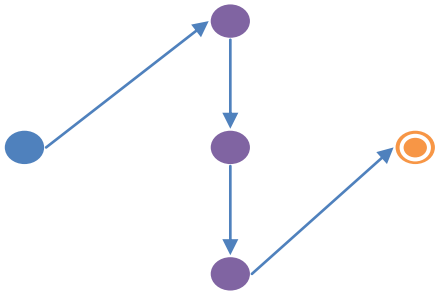

Fig. 3. Hidden Layer with cascading functions and with Single node Input and Output

Hence, instead of having every node supplying the input parallelly, we get a sequential flow of input and weight, which is passed to every function. Now every function further compute that input and sends the output to the green node. The job of the green node is to sum the outputs. Now, the objective is simply to determine the highest output from the sequential flow of input [10]. Since inputs are real numbers, we do not need to worry about inputs that do not follow this path as there will always be an input that fits in the real field [5]. The activation value of previous node is forwarded to the next node and the process continues.

IV. WEIGHT ADJUSTMENTS

Neural networks can readjust their weights while training [11][12]. An output is validated with the real output and the error calculated. The backtracking involved requires error estimation to be sent to the previous nodes and recompute node error [7][11]. Following this a weight re-estimation is done. To do the same for the given model, we can compute it as follows.

The final output Y for single (x_i,w_i) we get the error as the difference (Y – F(X*W)). This value F(X*W) is provided by the last hidden node just before giving the output. The Error is calculated and sent back to the previous Node. The node calculates the new Error and adjusts its output and sends back the Output error and node error to previous. This continues till all the nodes are updated with the error values and then a forward pass is done again to re-update the weights based on the new pass. Hence, the representation for backpropagation can be shown as the following fig. 4.

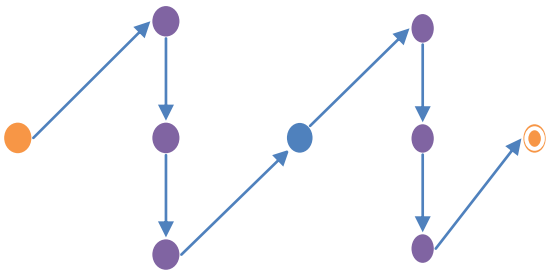

Fig. 4. Implementation of Backpropagation Algorithm in Architecture

The above diagram in Fig. 4 represents that the output calculates the error and forwards to last function that sent the cumulative output. The function calculates the error on the node and updates the output. The functions send the global error and self-node error in a cascading manner till input node is returned. Following which a forward pass is again tested to readjust the weights. As the weights are adjusted, simultaneously new values are produced and sent forward. Hence, forward pass is repeated and process is continued till error is lower.

V. DEEP NETWORK ANALYSIS

Deep networks are networks with more than one layer of hidden layers [11]. Since we have already established a Feed forward architecture, to convert it to its deep form is relatively easier as we have to just increase the number of functions that work with the input. Therefore, we come to the following for deep Feed forward architecture. The general architecture of deep neural network is shown in Fig. 5

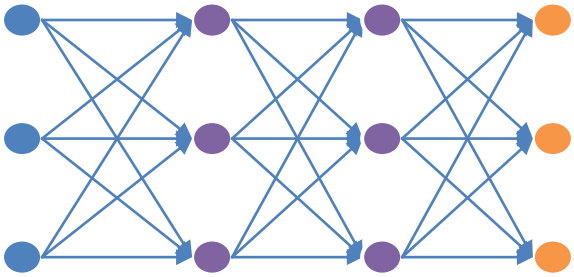

Fig. 5. Standard Deep Feedforward Network

VI. SKIP CONNECTIONS

For the above-mentioned models, we have assumed fully connected architectures. However, in many real-life cases there may be the introduction of inputs with function that are not connected as in the case of sparse interconnections. So, for those cases must look at skip connection that allows a function to skip computation and forward to the next node. The objective of the skip connection is to get previous value of hidden layer and forward that value to next function. The question arises as to why even keep such a node since its objective is to act like a bridge. Skip connections are input dependent and hence based on the model that needs to be followed require either act as a functional node or a bridge. The implementation can be done periodically, pre-determined or stochastically.

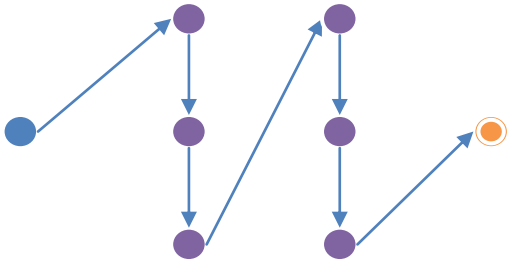

Fig. 6. Corresponding Sequential Representation of the deep network

VII. CONCLUSION

Therefore, we come to the following estimations for converting a convolution feedforward network to its estimated sequential counterpart. We have discussed a comparative analysis while deriving some key aspects of automata to imagine the same. However, it must be mentioned that the above estimation takes care of only feedforward architectures and no other architecture like RNNs, LSTMs or Gated Networks and others. Convolution neural networks are of prime interest and that consideration is only taken. Minor alterations can made again to allow for different variations of the same. Embedded architecture with time does increase performance and allow more power per device space but we consider small devices of low-end devices rather than costly integrated circuits. Hence, circuits for this proposal are expected to be really cheap hardware.

REFERENCES

1. Yang, F., & Paindavoine, M. . Implementation of an RBF neural network on embedded systems: real-time face tracking and identity verification. IEEE Transactions on Neural Networks, 14(5), 1162-1175, 2003.
2. Malinowski, A., & Yu, H. Comparison of embedded system design for industrial applications. IEEE transactions on industrial informatics, 7(2), 244-254., 2011
3. Jiao, L., Luo, C., Cao, W., Zhou, X., & Wang, L. Accelerating low bit-width convolutional neural networks with embedded FPGA. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL) (pp. 1-4). IEEE, 2017.
4. Han, S., Pool, J., Tran, J., & Dally, W. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems (pp. 1135-1143)., 2017.
5. Reyneri, L. M., Chiaberge, M., &Zocca, L. CINTIA: A neuro-fuzzy real time controller for low power embedded systems. In Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems (pp. 392-403). IEEE.,1994
6. Moini, S., Alizadeh, B., Emad, M., &Ebrahimpour, R. A resource-limited hardware accelerator for convolutional neural networks in embedded vision applications. IEEE Transactions on Circuits and Systems II: Express Briefs, 64(10), 1217-1221, 2017.
7. Bettoni, M., Urgese, G., Kobayashi, Y., Macii, E., &Acquaviva, A. A convolutional neural network fully implemented on FPGA for embedded platforms. In 2017 New Generation of CAS (NGCAS) (pp. 49-52). IEEE, 2017.
8. Iandola, F., &Keutzer, K. Small neural nets are beautiful: enabling embedded systems with small deep-neural-network architectures. In Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion (p. 1). ACM., 2017
9. Han, S., Mao, H., & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149., 2015
10. Farabet, C., Martini, B., Akselrod, P., Talay, S., LeCun, Y., &Culurciello, E. Hardware accelerated convolutional neural networks for synthetic vision systems. In ISCAS (Vol. 2010, pp. 257-260)., 2010
11. Canziani, A., Paszke, A., &Culurciello, E. An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678., 2016
12. Jian, B., Yu, C., &Jinshou, Y. Neural networks with limited precision weights and its application in embedded systems. In 2010 Second International Workshop on Education Technology and Computer Science (Vol. 1, pp. 86-91). IEEE,2010

AUTHORS PROFILE

Rajarajan G, currently working as Assistant Professor in the School of Computer Science and Engineering in Vellore Institute of Technology, Vellore. He completed his Ph.D in Anna University. His research domain includes Networks, Cyber Security, Computational Theory. He is a member of Indian Science Congress, IAENG.

Madhur Bhatnagar, currently pursiung his final year B. Tech CSE in Vellore institute of Technology, Vellore. He is a student member of CSI, IEEE. His area of interest includes Neural Networking, Machine Learning.

Sharoni Roy Chowdury currently pursiung her final year B. Tech CSE in Vellore institute of Technology, Vellore. She is a student member of CSI, IEEE. Her area of interest includes Neural Networking, Machine Learning, NLP.