A Novel Use of Deep Learning to Optimize Solution Space Exploration for Signal Integrity Analysis

Mruganka Kashyap
Department of Electrical and Computer Engineering
University of California, San Diego, USA
Email: mrkashya@eng.ucsd.edu
Phone: +1-858-291-2162

Dr. Kumar Keshavan
Cadence Design Systems
Chelmsford, Massachussets, USA
Email: ckumar@cadence.com
Phone: +1-339-221-2213

Dr. Ambrish Varma
Cadence Design Systems
Chelmsford, Massachussets, USA
Email: ambrishv@cadence.com
Phone: +1-978-262-6431

***Abstract*—The enhanced complexity of electrical devices has increased the number of variables that directly or indirectly affect the output. Consequently, it has become imperative to explore the massive solution space during integrity analyses, without sacrificing accuracy and development time. In this paper, we offer a simple hybrid algorithm based on a Multi-Layer Perceptron that significantly works better than traditional methods like Least Squares, by balancing the requirements for high accuracy and less development time.**

Keywords: *Macromodeling, Signal Integrity Analysis, Optimization*

I. INTRODUCTION

Advances in hardware technology for electrical devices has enhanced the quality of signal processing. But at the same time, there has been a significant increase in hardware complexity. This has resulted in an interplay of a large number of variables that directly or indirectly affect the output of the system to varying extents. There is also an increased presence of a significant number of redundant variables in relation to the output.
One of the strongest requirements in quality testing of electronic products is the need for exploration of design space of the product. But the massive computational complexity due to the presence of a large number of input variables, significant or otherwise, makes the traditional simulation based methods prohibitively time expensive. For instance, if a set of 4 input variables are considered, that may or may not be having an effect on a single output, and suppose, each of the input variable can take up to 50 possible values. This creates a requirement for approximately 6.25 million simulations to map out the entire design space. A single simulation with four variables for a single output, in the case of a simple topology like a DDR4 Multi Drop topology, takes an average time of 600 seconds on a 64-bit machine. Essentially, that creates a massive time requirement of 3.75 billion seconds, approximately 43,403 days. This paper explores a method to perform the mapping of the entire design space of a DDR4 interface [1].

A. Linear Least Squares Method (LLSM)

The standard approach often used to address this problem is the Linear Least Squares Method. The core objective of the method is to adjust the parameters of a model function to obtain the best data fitting for a given data set. This method is, however, biased due to the fact that the underlying model is often considered linear [2]. This is often an impediment in characterizing an accurate model for a collection of input variables in relation to the output variable, in most practical situations. A further variation of least squares method is the Nonlinear Least Squares Method. However, this method is fraught with problems of non-convergence [3]. For instance, in the DDR4 multi-drop topology, the prediction accuracy hits a maxima of 94% for the output variable, 'eye_njn', which is defined as the normalized jitter and noise and is computed by dividing the noise area of an eye by the total eye.

B. Artificial Neural Network (ANN)

Artificial Neural networks were inspired by biological neural networks in animal brains. It is based on a collection of connected units defined as artificial neurons. The neurons are arranged in layers and the training dataset is used to train the ANN, generating weights. The validation dataset uses the weights for prediction and based on the error in prediction compared to the true value, the ANN is trained further and the model is refined. The backpropagation algorithm accelerated the training of multi-layer networks and is now the generally accepted method for training [7]. Training is complete once the convergence criterion has been achieved and the model weights obtained are used to predict the outputs for the test dataset.

C. DataSet

Cadence® Sigrity™ SystemSI™ tool is used to generate the dataset by random sampling of the input variables. This allows the generation of a dataset that spans the entire solution space for the inputs and thus the entire design space. Six input variables, four of which are defined by length from Dimm2 to Dimm1 blocks (say len_dimm2dimm1), length

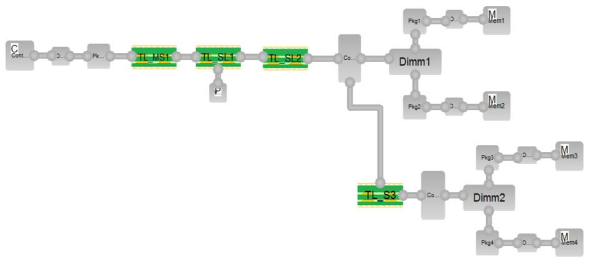

Fig. 1. DDR4 Multi Drop topology

from Dimm2 to Dimm2 (say len_dimm2dimm2), length of 40Ω impedance (say len_z40_ohm), length of 60Ω impedance (say len_z60_ohm), which are related to the output 'eye_njn', previously defined and two other dummy variables, are considered. Each of the input variables can take 20, 20, 20, 6, 6 and 50 distinct discrete values respectively. The total number of combinations to map out the entire design space is 14.4 million. A 1000 random sampling based simulation generated dataset is obtained to serve as the preliminary data.

II. Hybrid Algorithm

1) Cross-Correlation: Cross-correlation is defined as the measure of similarity of two data series as a function of the displacement of one in relation to another. The first step in the hybrid algorithm proposed is to attempt a substantial reduction of the solution space. This includes determining the variables that actually affect the output and discarding the redundant input variables. For the dataset, cross-correlation per input variable in relation to output is defined by:

$$r = [\Sigma\Sigma(x - \mu_x)(y - \mu_y)]/[(n-1) * \sigma_x * \sigma_y] \qquad (1)$$

An input variable with a cross-correlation value within the range of $[-a, a]$, where $a = 0.1$ on a preliminary dataset is considered as having negligible effect on the output, 'eye_njn' and is discarded. The cross-correlation values are 0.50869967, 0.2028228, 0.22503745, 0.31313376, 0.04023583 and 0.02123405 respectively for the the input variables in order. The values of 0.04023583 and 0.02123405 are discarded and incidentally, they are the cross-correlation values corresponding to the two dummy variables. Besides that, we can also reasonably conclude that the variables 'len_dimm2dimm1' and 'len_z60_ohm' are more positively correlated with the output and thus, the major input variables, than the other two variables.

2) Multi-Layer Perceptron (MLP): The removal of 2 variables reduces the solution space by 300. A secondary random sampled simulation generated dataset with 595 data points is generated. As a sanity check, a cross-correlation for the remaining 4 variables is conducted and the values are 0.32768618, 0.02140264, 0.02482475 and 0.10532641 in order. This is concurrent with our previous deductions from the preliminary dataset. This data serves as the input for the MLP.

An MLP factors in any non-linear relationship between the input variables and the output and is more robust compared to the LLSM [4]. It is a feedforward artificial neural network mapping a set of input data to the output data [5]. It consists of multiple layers of nodes, called neurons, with each layer fully connected with the subsequent one. Each neuron, except for the input nodes is a processing element and is defined by an activation function. Nonlinear activation function like the 'hyperbolic tan' function is used in the hidden layers to model the frequency of action potentials of biological neurons in the brain. Even though the sigmoid activation is much more biologically realistic, the tanh function is symmetric about the origin [5]. The MLP implemented in this algorithm consists of an input layer, consisting of 4 neurons, equal to the number of input variables. It has atleast 2 hidden layers with number of neurons equal to 10 in each. This is done to allow incorporation of reasonable regularization through Dropout in each layer [6]. Each of the hidden layers is followed by Dropout Layers with a dropout factor of 50%, to prevent overfitting during the training of the MLP. A similar dropout of 20% is incorporated after the input layer as well. The final output layer consists of a single neuron equivalent to the number of output variables. The output neuron provides the final value of the output. Learning occurs in the MLP by altering connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is a supervised learning technique called backpropagation for training the network [5]. The Mean Squared Error is used as the loss function during training and validation.

The 595 data-points dataset is divided into 150 for testing, 395 for training and 50 for validation. The 395 datapoints are used for training the MLP, in order to obtain a model that defines the relationship between the inputs and the outputs. An epoch is a measure of the number of times all of the training vectors are used once to update the weights. The convergence occurs when the average absolute error between predicted and actual outputs over 3 epochs is less than a tolerance value, in this case, of 0.02. Over 100 runs of the MLP, convergence occurs at an average of 200 epochs. Training accuracy reaches an average of 97.5% with a 1σ of 1.6%. The validation dataset is used to check the effectiveness of the model after training in each epoch. Validation accuracy reaches an average of 97.4% with a 1σ of 1.6% over 100 runs of the MLP.

Validation of the model, defined as testing, generated post-training, after completion of all epochs, was also done by predicting the output variable (for example, eye height/width or eye_njn) of a random simulation run and comparing the predicted value with the actual simulation result with the same inputs.

III. Final Results

Fig. 2 depicts a plot of the validation accuracy with the increase in epochs. It can be easily observed that the plot

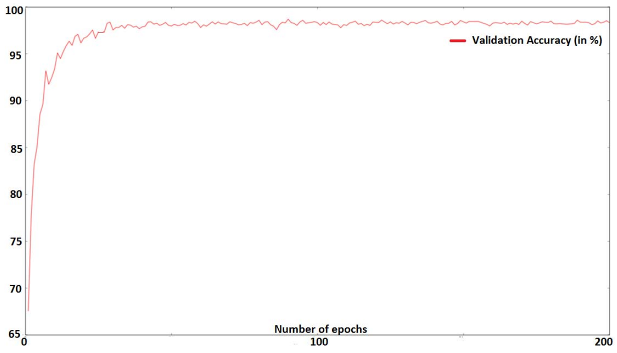

Fig. 2. Plot of validation accuracy vs Number of epochs

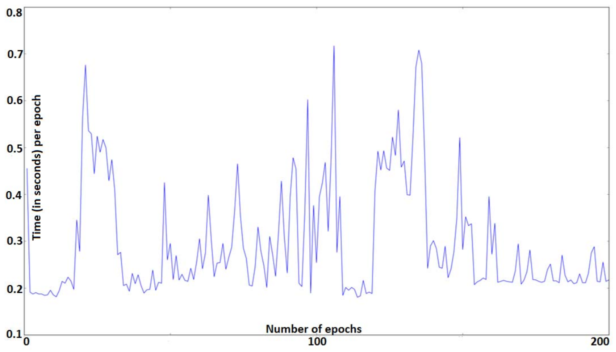

Fig. 3. Training time vs Number of epochs

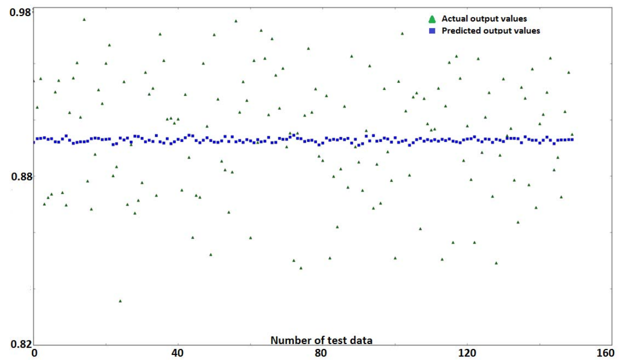

Fig. 4. Predicted and actual outputs vs Sample number

begins to converge as we approach towards 200 epochs. As can be observed in Fig. 3, the total average time taken for each epoch of 0.5 seconds, even by conservative estimates, yields an average total training time of 100 seconds. The time taken for testing is a maximum of 20 seconds for 150 data points. By and large, the time consumed in implementation of this algorithm is due to the 1000 preliminary data and 595 secondary data generation. This is significantly less than the total simulation time for 14.4 million combinations to map out the entire design space. Fig. 4 represents the actual outputs in green triangles and the predicted outputs of the algorithm in blue squares.
The mean for the testing accuracy on a 150 test dataset is 97.3% with a 1σ of 1.8%. This is considerably better than the test accuracy obtained by modelling using the LLSM.

IV. Conclusion

The current method used for mapping the design space involves brute force sweep to cover the solution space. It is often a non-starter for any serious analysis, requiring a new methodology for analyzing large design space. A simple Hybrid Methodology was devised to determine the minor and major variables with a small random sampling of the solution space by balancing high accuracy and less development time. Following removal of the redundant and minor variables, intelligent data is generated. This intelligent data is used for training and validation of an Artificial Neural Network. This Artificial Neural Network based on a Multi-Layer Perceptron can be used as macro-model to predict output variables in lieu of simulation, thus, increasing options for design space exploration. This method can be used for prediction of multiple outputs across different datasets with almost little or no loss in prediction accuracy. A further extension of this procedure is suggested for reduction of the range of values from the input variables; thus, a simple plot between the output and input variables can be obtained and then the ranges only correlated could be considered as valid.

Acknowledgment

The principal author would like to thank Dr. Kumar Keshavan and Dr. Ambrish Varma for their valuable insights and encouragements. The authors would also like to express gratitude to Cadence Design Systems, Chelmsford for making this project a success.

References

[1] C. D. Systems. (2014, Oct 15). *Cadence Announces Industry's First Multi-Protocol DDR4 and LPDDR4 IP Solution*. [Online]. Available: https://ip.cadence.com/news/504/330/Cadence-Announces-Industry-s-First-Multi-Protocol-DDR4-and-LPDDR4-IP-Solution.
[2] D. G. Luenberger, "Least-Squares Estimation," in *Optimization by Vector Space Methods*, New York: John Wiley & Sons, (1997) [1969],pp. 78-102.
[3] C. T. Kelley, "Iterative Methods for Optimization," in *SIAM Frontiers in Applied Mathematics*, 1999, vol. 18.
[4] G. Cybenko, "Approximation by superpositions of a sigmoidal function," in *Mathematics of Control, Signals, and Systems*, 1989, vol. 2.4, pp. 303314.
[5] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice Hall, 1998.
[6] Nitish Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," in *The Journal of Machine Learning Research*, 2014, vol. 15.1, pp. 1929-1958.
[7] Paul John Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Applied Mathematics, Harvard University, MA, 1974.