Using Cell Convolutional Neural Network for Classification of VLSI Cells

Rohit Sharma, rohit@paripath.com* *Paripath Inc*

Abstract: Identification of Very Large Scale Integrated (VLSI) cells using graph matching is a NP-complete problem. We propose a pioneering approach of converting graph isomorphism problem into deep convolution neural network (CNN) with VLSI cells identification as a case study. This proposal defines cell convolution as a method to identify VLSI cell [16]. It uses this approach to classify VLSI cells by translating cell graph into a numerical matrix and feeding it to a CNN. This work used fully extracted circuit netlist of over 11,000 cells from few large cell libraries with variety of architecture and functionality. Input data was conditioned before being fed to a carefully crafted convolutional network with an objective of achieving over 90% accuracy while keeping computational budget to a minimum. These matrix permutations range from few billions to over googols of googol (order of 10^{900}). Approach of identifying VLSI cells using deep convolution neural network is pioneering in the field of VLSI design automation and can be applied to its sub-domains including subgraph matching in layout versus schematic checking, partitioning of larger circuit with cell identification among others.

***Index Terms*—Very Large Scalable Integration, Cell Classification, Convolutional Neural Networks, Cell Convolution**

I. Introduction

In the last five years, deep learning frameworks have enjoyed phenomenal success in vision, video, speech and text processing. This phenomenon has catalyzed exploration of deep learning capabilities to many other domains including finance, healthcare, defense to name a few. Deep neural networks often use convolution primarily because of their offerings in translation invariance and local feature extraction. These two offerings are useful in many areas of VLSI design automation industry; however, the industry has largely ignored advances in deep learning primarily because of statistical nature of methods and results, and lack of clear value proposition. Last time industry tried replacing deterministic approach to statistical one in timing sign-off, it failed miserably in changing well stablished sign-off methodology [1]. Other challenges remain suitability of non-numeric input data, extremely large size of input data and continuously changing requirement of evolving process nodes. However, Continued success of deep learning has inspired us to explore its application in design automation.

In this paper we define VLSI cell convolution in order to reduce graph matching problem to a convolution neural network problem. In doing so, we share our challenges in designing a system, codenamed NEO, capable of classifying VLSI cells using neural network. In section III, we present the idea of cell convolution. Section IV describes problem of cell classification and prior art. Section V outlines challenges associated with input data gathering, augmentation and conditioning. Section VI presents cell convolution neural network (CCNN) architecture. In section VII, we outline training methodology of proposed neural network. Section VIII talks about setup and results. Conclusion is presented in section IX.

II. Related Work

Starting with success of LeNet-5, deep convolutional neural networks have enjoyed continuous attention resulting in improvement of capabilities including algorithms, network architecture and results thereof [3]. Popularity of deep convolutional neural networks peaked since the advent of ImageNet Large Scale Visual Recognition Challenge (ILSVC) [4]. In last 5 years (2012 to 2017), the top-5 error rate for the ILSVRC competition has dropped from 26.2% to 3.57%, just another testament to the popularity of CNNs.

Neural Networks in VLSI design automation, however, is a relatively new concept and struggling to find a successful application. University of Illinois Urbana-Champaign research center for advanced electronics through machine learning [5] is developing new domain-specific machine learning algorithms to extract models. They have 6 different projects in progress as of late 2017. We've contemplated few ideas to incrementally improve user experience in different sub-domains of VLSI design automation. Logic simulation and verification flows can benefit from multinomial logistic regression [6] to automate the test-bench generation process for faster results. Another applicable field is behavior modeling in simulation. Convolution Neural Network is also shown to learn to route a circuit layout net. [8] Initial applicable work on graphs with neural network was reported in 2009 [2]. This work successfully showed one subgraph matching at a time as an application, which is incredibly useful for VLSI Design Rule Checks. No other VLSI design automation research is

known to use convolution neural network approach at this time.

III. Cell Convolution

Here we present main idea of cell convolution using a simple AND gate (example cell) consisting of NAND gate followed by INVERTER gate in CMOS technology. Transistor level representation of the cell is shown in Figure III-1.

Table 1 shows the connectivity of the cell in the form of a matrix of numbers suitable for performing convolutions. First column of this matrix shows PMOS and NMOS transistors represented as arbitrary integers 20 and 10 respectively. Rest of the 3 columns in the matrix are drain, gate and source nodes of MOS devices with numbers as marked in the Figure III-1.

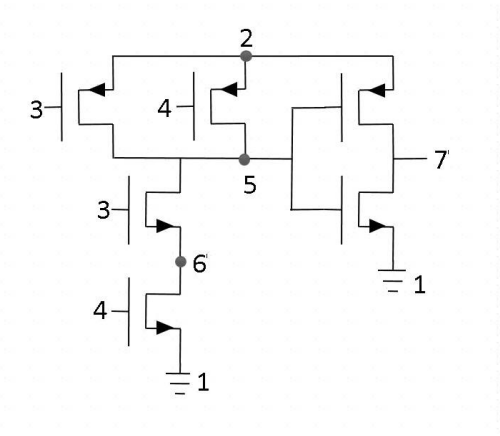

Figure III-1Transistor level representation of an example cell

*MOS	DRAIN	GATE	SOURCE
20	2	3	5
20	2	4	5
10	5	3	6
10	1	4	6
20	2	5	7
10	1	5	7

Table 1: cell's netlist represented as matrix

1	1	1	1
-2	-2	-1	-1

Table 2: Convolution filter to detect inverter

-3
-8
0

Table 3: Convolution Layer detecting inverter as 0

Table 2 is simple convolution filter of size 2x4 designed to identify INVERTER gate. Table 3 is resulting convolution layer obtained by applying convolution filter of Table 2 using a stride of 2 with valid padding onto AND matrix of Table 1. Zero in the last row of resultant convolution layer represents successful identification of inverter (similar to cost function in neural network) present in last two rows of the matrix in Table 1.

This example paints a simplified picture of cell identification through convolution. There are several challenges in putting this to practice including non-uniformity of data, ill conditioning and augmentation, large number of permutations a circuit can be represented as a matrix using numbers and tuning the hyper parameters of neural network.

A. Matrix Permutations

It is noteworthy to mention astronomically large number of permutations a VLSI cell's electrical network (like the one shown in Figure III-1) can be represented as a matrix. Inspired by M. Ohlrich's work [9], we reserve node number 1 and 2 for special nodes ground and power respectively. Rest of the nodes can be labelled arbitrarily without changing the electrical network.

Let's assume that n is number of nodes and d is the number of devices in the electrical network. Number of permutations an electrical network can be labelled is given by equation below:

$$(n-2)! \tag{1}$$

Expression above has 2 is subtracted because of 2 special nodes ground/power are nonpermutable.

Now that nodes are labelled, they are placed with devices shown in Table 1 consistent with SPICE format [12]. Permutations devices can be ordered in the matrix is given by:

$$d! \tag{2}$$

Given drain and source terminals of any device (matrix row) can be interchanged without changing electrical network, total number of combinations for this operation is given as:

$$\sum_{i=1}^{n-2} d!/(i!.(d-i)!) \tag{3}$$

From (1), (2) and 3, total number of permutations an electrical network can be represented as matrix is

$$d!(n-2)!.\sum_{i=1}^{n-2} d!/(i!.(d-i)!) \tag{4}$$

Let's compute number of matrix permutations for our example in section III. With n=7 and d=6, total number of matrix permutations are just under 14 billion. For one of the larger sized cell in the dataset with (d=) 351 devices and (n=) 44 nodes, matrix permutations are over $1e^{900}$.

Convolutions have well recognized properties of feature localization and translation invariance. It is tempting to use them to reduces the requirement on the permutations a cell graph ought to be represented as a numerical matrix.

IV. VLSI Cell Classification And Identification

In this paper we use the term classification and identification interchangeably, since former can be described as superset of later. When a unique label is assigned to each member of the class, problem of classification reduces to that of identification. For example, a n-input-1-output AND gate class may include many variations including number of inputs, transistor count, transistor strengths, planer vs 3D etc.; they all are classified as AND gate. When each of these variations are assigned a unique label, cell classification problem reduces to cell identification.

VLSI cell identification problem is a NP-complete problem. Several heuristic solutions using subgraph isomorphism have been proposed in last three decades [9], [10] and [11]. These solutions suffer from unpredictable nature of performance and do not guarantee solution. These two imperfections make VLSI cell identification problem a good candidate for convolution neural network approach.

V. The Dataset

We used fully extracted circuit netlist of over 11,000 cells from few large cell libraries with variety of architectures and nanometer process nodes as our dataset. Parasitic (resistance and capacitance) of the VLSI-cell are undesired elements (like noise in the image). Parasitic add to significant size (typically 10 times) to dataset and do not define or contribute to feature set of the class, so they were removed. This reduced our matrix size by a factor of 10. Our analysis revealed that 4th terminal (called substrate) had strong correlation with device type (NMOS, PMOS etc.) and therefore was a good candidate for removal from cell matrix. However, it appeared to impact training accuracy for a certain class adversely; and therefore, it was kept in the matrix. Inspired by the prior works in subgraph matching we added one additional feature for every device two additional features for every node in the matrix [9], [11]. With this modification, every device row in the matrix has 13 additional columns. Size of resulting matrices were adapted to fit predetermined input size of convolutional neural network

Cells were assigned total of 6 labels based on their functional similarity with no regard to structural or graph attributes. 80 percent of this set was used in training the neural network and 20 percent was reserved as a test set.

VI. Cell Convolution Neural Network

Our proposed model Cell Convolution Neural Network (CCNN) was designed with a goal to achieve over 90% accuracy while keeping computational budget minimum. Computational budget included a basic quad core machine with 16G RAM. Computational budget forced us to make most of our CCNN parameters decisions like number of layers, filter size, pooling etc.

Figure VI-1: Cell Convolution Neural Network

Layer no.	Layer Type	Filter size/stride	Output size
1	Convolution	2x2 / 1	369x14x16
2	ReLU	-	
3	Convolution	2x2 / 1	369x14x32
4	ReLU	-	
5	Convolution	2x2 / 1	369x14x64
6	ReLU	-	
7	Flat	-	330624
8	Fully Connected	-	6x1
9	SoftMax	-	6

Table 4: Cell Convolution Neural Network Parameters

Original VLSI cell sizes ranged from 2x14 to 369x14, which were adapted to 369x14. No scaling methods, in an attempt to keep cell-size uniform, led us to retain enough information to achieve desired accuracy. So full sized 369x14 matrices were used.

As shown in Figure VI-1, CCNN has a total of 9 layers. First 6 layers are convolution layers followed by rectified linear units (ReLU). Uniform convolution filters of size 2x2

with stride of 1 and valid padding were used for all three convolutional layers. Last 3 layers are flat layer, fully connected layer and SoftMax layer.

CCNN layers sizes are shown in Figure-VI-1 and again repeated in table 4. SoftMax layer is used to produce compute cell-labels, cost function and accuracy. Pooling was not used because of its adverse effect on accuracy. Regularization, although experimented with, wasn't required because of absence of overfitting. We omit the rest of the details of the network, as empirical evidence suggests that the influence of the exact parameter choice is minor.

VII. Training

We used a basic quad core machine with 16G RAM to train our CCNN. Our training used Adam Optimizer [13] with a learning rate of 5e-5. We found the suggested learning rate of 0.001 causing severe overfitting with a gap of over 20% between training set and test set. We used logarithmic search to fine tune learning rate of 5e-5. we found leaving the momentum term β_1 at the suggested value of 0.9. We experimented with mini-batch size and max pooling to maintain RAM requirement and found mini-batch size of 512 with no pooling for training more effective than other combinations. We followed training tips including normalizing training/test set, initializing weights/biases, mini batch size and tuning other hyper parameters mentioned in the book by I. Goodfellow, Y. Bengio et. al [14].

CCNN was optimized for 166 epochs with a total of 3000 iteration with a minibatch size of 512 on a training set of over 9000 cells with matrix of 369x14 each. It took over 32 hours to train this network.

VIII. Results

Results of training the network are shown in FigureVIII-1 and Figure VIII-2. At the end of last epoch, CCNN training and validation accuracy were recorded as 90.5% and 89.3% respectively as shown in Figure VIII-1 Maximum training and validation accuracy were recorded as 91.1% and 89.5% respectively in 157th and 166th epoch. Accuracy graph shows instability during training which it is able to recover from. This instability has our curiosity spiked and has been reserved as a task to iron out in future.

CCNN training and validation loss is also shown in Figure VIII-2. As expected, loss curves follow a similar trend as accuracy curve.

A. *Future Work*

Although we reached our target of 90% accuracy, it is by no mean an ultimate result. Cell convolution concept deserves more effort on advancing accuracy by experimenting with deeper architectures and hyper-parameters with more resources including variety of input data and a large computational budget.

While this work was focused on identification of VLSI cells, it is applicable to every problem solved using graph and subgraph isomorphism

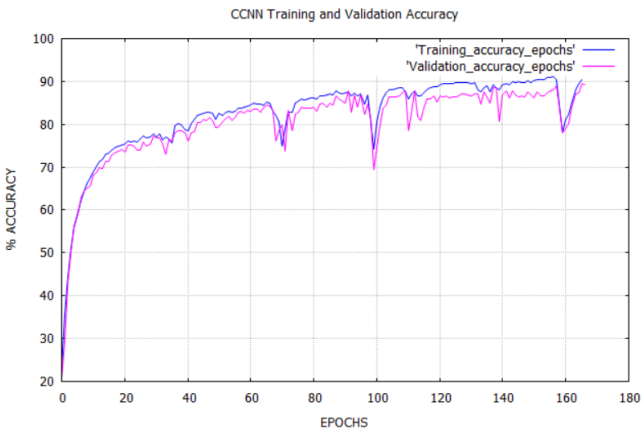

Figure VIII-1: CCNN Training and Validation Accuracy

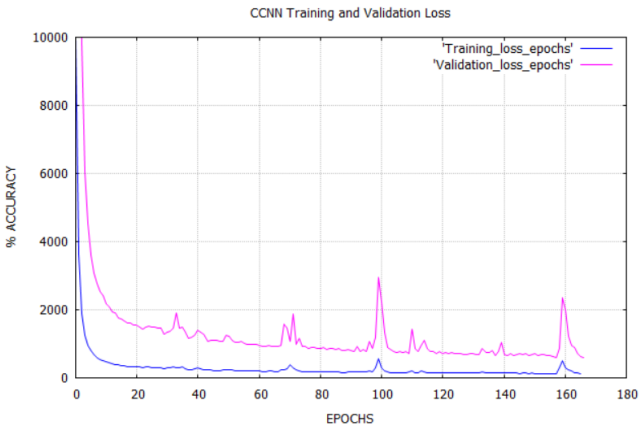

Figure VIII-2: CCNN Training and Validation Loss

Initial success of CCNN has opened up several avenues for CNN in graph matching in general and design automation space in particular. It is tempting to use Region based CNN (RCNN) to apply in subgraph matching in general and partitioning large IPs consisting of VLSI cells in design automation in particular [15]

IX. Conclusion

In this paper, we reduced graph matching problem to a CNN problem using VLSI cell classification as a case study. We also showed that CCNNs have potential to serve as effective alternate solutions to cell classification or cell identification problem in design automation.

References

[1] M. Orshansky and A. Bandyopadhyay. Fast Statistical Timing Analysis Handling Arbitrary Delay Correlations. In Proc. ACM/IEEE DAC, pages 337–342, 2004.

[2] Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M. & Monfardini, G. 2009, The graph neural network model, IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989

[4] ImageNet: A Large-Scale Hierarchical Image Database Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei Dept. of Computer Science, Princeton University, USA {jiadeng, wdong, rsocher, jial, li, feifeili}@cs.princeton.edu. http://image-net.org

[5] UIUC Center for Advanced Electronics Through Machine Learning (CAEML) https://publish.illinois.edu/advancedelectronics/research/
[6] Multinomial Logistic Regression, Nursing Research: November-December 2002 - Volume 51 - Issue 6 - p 404-410
[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions, In Computer Vision and Pattern Recognition (CVPR), 2015.
[8] Jain, Sambhav R., and Kye Okabe. "Training a Fully Convolutional Neural Network to Route Integrated Circuits." arXiv preprint arXiv:1706.08948 (2017).
[9] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, "SubGemini: identifying subcircuits using a fast subgraph isomorphism algorithm," Proc. 1993 Design Automation Conference, pp. 31–37, June 1993.
[10] A. R. Conn et al., "Gradient-based optimization of custom circuits using a static-timing formulation," in Proc. Design Automation Conf., June 1999, pp. 452–459
[11] J. Larrosa and G. Valiente, Constraint satisfaction algorithms for graph pattern matching, Math. Struct. Comput. Sci. 12 (2002) 403–422.
[12] L. W. Nagel and D. O. Pederson "Simulation Program with Integrated Circuit Emphasis" Proc.Sixteenth Midwest Symposium on Circuit Theory, Waterloo, Canada, April 12, 1973.
[13] Kingma, Diederik P and Ba, Jimmy Lei. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[14] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.
[15] Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR, abs/1311.2524v5, 2014. Published in Proc. CVPR, 2014.
[16] Rohit Sharma, 2015. Characterization and Modeling of Digital Circuits, ISBN: 9781500733674

Rohit Sharma Rohit Sharma is an electronics and computer engineer. He holds a master degree from Indian Institute of Technology, Delhi, He has published several papers in international conferences and journal and continues to contribute to electronic design automation domain for over 19 years learning, improvising and designing solutions. He is passionate about many technical topics including characterization, modeling and analysis. He currently works for Paripath Inc. (paripath.com) - a company he founded.