ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0066

Sum-Product Network, Extension into Discovery of Karatsuba Multiplication

Zhiyang Wang[1], Hongbo Zhang[2], Tingfeng Huang[3], Yuang Zhou[4], Haihang Peng[5]

[1]School of Science, Simon Fraser University, Vancouver, V5A1S6, Canada;

[2]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876, China;

[3]International School, Beijing University of Posts and Telecommunications, Beijing, 100876, China;

[4]School of Electronic and Electrical Engineering, Shanghai university of Engineering and Science, Shanghai, 201620, China;

[5]Jinan University - University of Birmingham Joint Institute, Jinan University, Guangzhou, 510632, China.

Abstract

The outstanding performance of neural networks usually comes at the expense of increased computational cost and hence demands improved efficiency. Among those, the sum-product network (SPNs) takes the spotlight. It disentangles multiplication and addition operations to permit a budget on the number of its multiplication operations, thus rendering fewer computing steps involved. Upon the budget constraint and supply of training sets, the network can be challenged to rediscover the Strassen multiplication algorithm with fewer number of scalar matrix multiplication. Given the importance mutually shared by Karatsuba algorithm and Strassen algorithm, the paper proposes that the application of the two-layer sum-product neural framework can be extended into discovering Karatsuba's integer multiplication algorithm by forcing it to learn with fewer multiplication operations.

Keywords

Sum-Product Network; Strassen Multiplication; Karatsuba's Algorithm.

1. Introduction

The deep neural networks, when given considerable amount of training sets, have illustrated the outstanding performance in predicting the reliable results, even though such outcome often comes at the expense of large model size and corresponding computational inefficiency [1]. As a result, the drawbacks make researchers investigate new algorithm design principles that could lead to improved computing efficiency. One prevailing principle is the reduction of the number of multiplications [1]. From the hardware perspective, trading multiplications against additions is desirable, since fewer number of multiplications elicits benefits in consuming less energy than that of additions does[2]. Further, addition/subtraction can be encoded as binary weights that can reduce the physical model size, and that be used in memory-bound deep learning application [3].

In terms of the principle's application in neural networks, researchers have attempted to test its efficacy in one prominent example -Sum-product network (SPN). By limiting the number of multipliers, Elser [4] forces SPN to discover the Strassen multiplication rules that adopt 7 instead of 8 scalar multiplications to compute two 2x2 matrices. In addition, Tschannen [1] constructs a novel network that casts matrix multiplications as 2-layer SPN and that is able to rediscover Strassen's multiplication algorithm, eliciting further potential applications.

Given the procedural similarity to Strassen's algorithm, it is surprising that no prior work has been done relating to the discovery of the Karatsuba algorithm with the powerful SPNs. Therefore, the aim of this paper focuses on the extension of SPN application from discovery of Strassen multiplication to exploration of Karatsuba integer multiplication. Further, the paper proposes that the training of a 2-layer SPN structure can deliver the prominent discovery of Karatsuba's algorithm.

2. Literature review

2.1 What is Sum-Product Network (SPN)

Sum-product networks (SPN) are a class of deep probabilistic models praised to provide tractable representations of probability distributions. The construction of this newly developed deep architecture is meant to reduce the complexity of the partition function to better obtain results of graphical model inference and learning [5]. In general, its major structural difference presents the benefit of simplicity and accessibility. Such network is composed by sum nodes and product nodes, of which the former corresponds to mixtures over subsets of variables and the latter refers to features or mixture components. All these variables produce internal linkage by weighted edges, to which transforms realized by the SPNs are adapted [1].

The prominent advantage of SPN over other networks is that inference in SPNs is guaranteed to be tractable, since inference, as a subroutine of learning, in expressive probabilistic models is generally intractable, which makes them difficult to learn and hence limits their applicability [6]. The structure and parameters of SPN that have layers of hidden variables admit its learning effectively and accurately from sufficient data [6]. It is shown that the improved performance of inference in SPNs in turn makes learning faster and more accurate than in previous deep networks [5].

On the evaluation of image completion supplemented by Caltech and Olivetti datasets, Lee et al [7] discovered that SPN successfully completed most faces by hypothesizing the correct locations and types of various parts like hair, eye, mouth, and face shape and color. In such process, they also observed that the learning iterations in SPNs are an order of magnitude faster than in deep networks such as Deep Boltzman Machines (DBM) and Deep Belief Networks (DBN). Previous studies have demonstrated the possible deployment of neural network for discovery of algorithms for shorter multiplications, potentially rendering much less time and fewer computing resources to be saved. Other explorations have indicated the applicability of the network. Adel [5] extended the SPN with the proposed generative and discriminative algorithms and found higher log-likelihood values and much faster performance on image recognition and digit classification when compared with other SPN algorithms.

Elser [4] presented interesting way in which neural networks are tested by challenging the networks to learn the rules of matrix multiplications, in accordance with Tschannen et al[2] view that matrix multiplications are the large fraction of the arithmetic operations used to evaluate deep neural networks. Elser's work alluded to a meaningful aspect that the architecture of SPN is a perfect match to solving mathematical problem as opposed to be a platform for general machine learning application. In the case of Strassen Multiplication (SM), standard algorithm returns 8 multiplications for the 2x2 matrices, yet the Strassen algorithm yields only 7, reducing the time complexity rom $O(N^3)$ to $O(N^{2.8})$. Further, Elser's work demonstrated the efficacy of the SPN in learning a correct set of weights by sufficient examples of correctly multiplied matrices.

The work, therefore, mandates that a proper construction of sum-product neural networks complemented with the protocol of conservative learning has the computational capacity for implementation of Strassen algorithm. Although the speed of learning and inference is more promising than the traditional deep network, much still remains unknown about its further applications [8], which ignites the passion to the investigation of this paper. Moreover, given that the SPNs enable researchers to impose a budget on the number of multiplication operations [1], the work encourages a promising direction to which other similar algorithms such as Karatsuba's Algorithm may be potentially discovered by the help of the network.

2.2 What is Karatsuba's Algorithm

The Karatsuba's algorithm (KA) is a multiplication algorithm published in 1962, whose efficiency in multiplication of two polynomials is of importance in applications like signal processing and coding theory [9]. It saves coefficient multiplications at the cost of extra additions, but is considered efficient since the total operating cost is less than the cost of the ordinary method [10]. Mishra [9] states that Karatsuba's multiplication algorithm uses three single-digit multiplications to perform one two-digit multiplication. The recursive application takes only 3^n single-digit multiplications to multiply a pair of 2^n-digit numbers. Hence, the work reduces the multiplication of two n-digit numbers with a time complexity to at most $n^{\log}2^3 = n^{1.58}$ single-digit multiplications in general (and exactly $n^{\log}2^3$ when n is a power of 2), as compared to conventional method that requires n^2 single-digit products. Paar [10] further indicates that while many algorithms have lower asymptotic complexity than KA, the latter one confers particular efficiency for multiplication of large numbers. Additionally, polynomials using KA requires fewer multiplications and additions combined.

The basic step of Karatsuba's algorithm permits the use to compute the product of two large number x and y using three multiplications of smaller numbers, each with about half as many digits as x or y, plus some additions and digit shifts. This basic step concludes generalization of a similar complex multiplication algorithm, where the imaginary unit i is replaced by a power of the base. Let x and y be represented as n-digit strings in some base B. For any positive integer m less than n, the two given numbers can be written as follows:

$$x = x_1B^m + x_0,\ y = y_1B^m + y_0, \tag{1}$$

$$xy = (x_1B^m + x_0)(y_1B^m + y_0) = z_2B^2m + z_1B^m + z_0, \tag{2}$$

$$z_2 = x_1y_1,\ z_1 = x_1y_0 + x_0y_1,\ z_0 = x_0y_0 \tag{3}$$

3. Methods

Karatsuba's algorithm and SPN application

What this research purports to accomplish is the discovery of Karatsuba's algorithm by training the proclaimed 2-layer SPN with a combination of datasets to learn the edge weights. With the proper initializing setup of the network, it is projected the network will soon arrive at the fewer number of combinations of multiplication when exposed to data of matrices. In addition to locate the values of weights, extra interest targets those weights that can return the fastest combination to do the multiplication in the multiplication of two numbers.

Before kickstarting the training process in SPN, figure had been vectorized into which could be applied in the Karatsuba's algorithm of great certainty. In general method, integer was used as input training data, so as to decompose numbers with powers of 10 and the result obtained a nx1 vector. Subsequently, we multiplied the vector with weight vector which consisted of 0 or 1 and then formed a (n+1) x n vector that would multiply the vector decomposed earlier from integer. This step theoretically laid out the foundation on which the training takes place to discover the Karatsuba's algorithm.

$$10^mx_0 + 10^nx_1 + x_2 \tag{4}$$

Then we should apply what we have done to Karatsuba's algorithm, we set the hidden layers to complete the function that Karatsuba's algorithm does, which means that we multiply (n+1)x(n+1) Karatsuba vector and (n-1)x1 decomposing vector in this step. The output of the whole training will be n different decomposed figure, what we will do is to use the power of 10 to restore original number. Then we successfully finish the product node setting and, in next step, we bring the sum nodes and product nodes in 2-layers SPN to carry on the training.

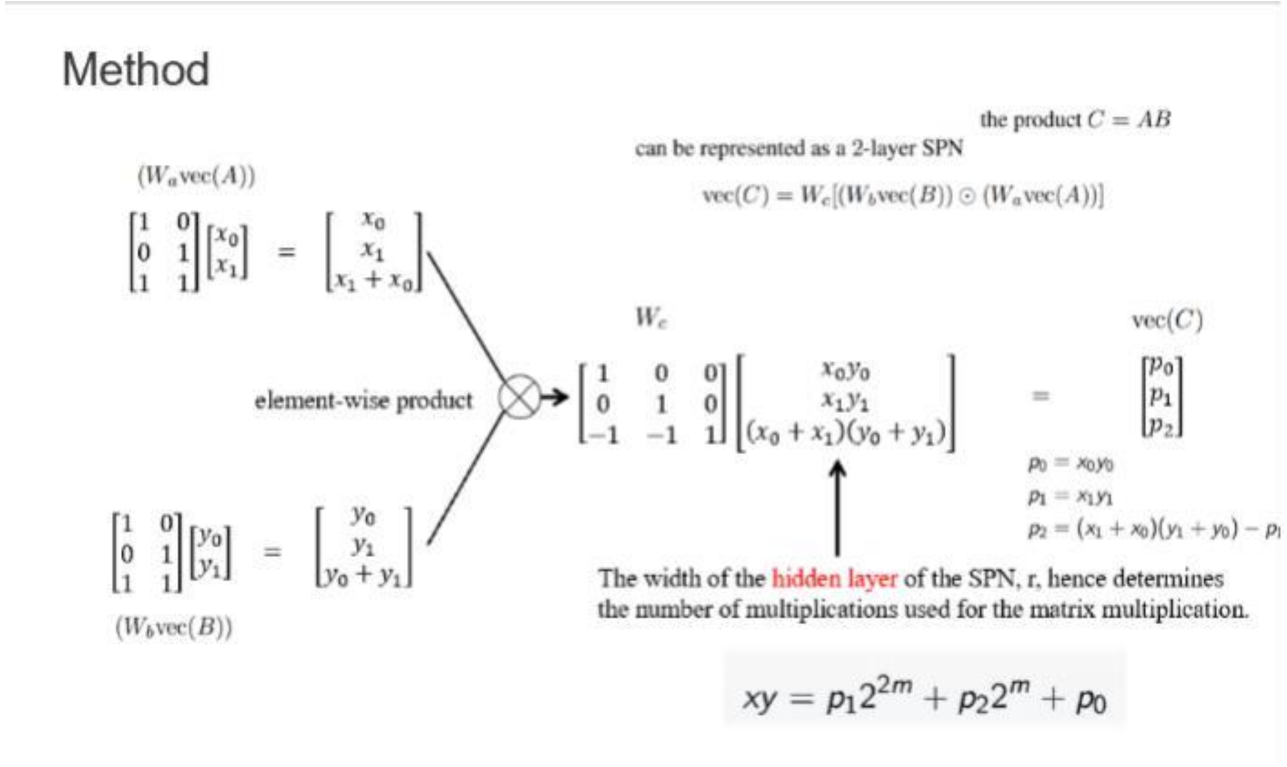
Method
the product $C = AB$
can be represented as a 2-layer SPN
$\mathrm{vec}(C) = W_c[(W_b\mathrm{vec}(B)) \odot (W_a\mathrm{vec}(A))]$
$(W_a\mathrm{vec}(A))$
$\begin{bmatrix}1&0\\0&1\\1&1\end{bmatrix}\begin{bmatrix}x_0\\x_1\end{bmatrix} = \begin{bmatrix}x_0\\x_1\\x_1+x_0\end{bmatrix}$
element-wise product
W_c
$\mathrm{vec}(C)$
$\begin{bmatrix}1&0&0\\0&1&0\\-1&-1&1\end{bmatrix}\begin{bmatrix}x_0y_0\\x_1y_1\\(x_0+x_1)(y_0+y_1)\end{bmatrix} = \begin{bmatrix}p_0\\p_1\\p_2\end{bmatrix}$
$p_0 = x_0y_0$
$p_1 = x_1y_1$
$p_2 = (x_1+x_0)(y_1+y_0) - p_1$
$\begin{bmatrix}1&0\\0&1\\1&1\end{bmatrix}\begin{bmatrix}y_0\\y_1\end{bmatrix} = \begin{bmatrix}y_0\\y_1\\y_0+y_1\end{bmatrix}$
$(W_b\mathrm{vec}(B))$
The width of the hidden layer of the SPN, r, hence determines the number of multiplications used for the matrix multiplication.
$xy = p_1 2^{2m} + p_2 2^m + p_0$

Figure 1: SPN network

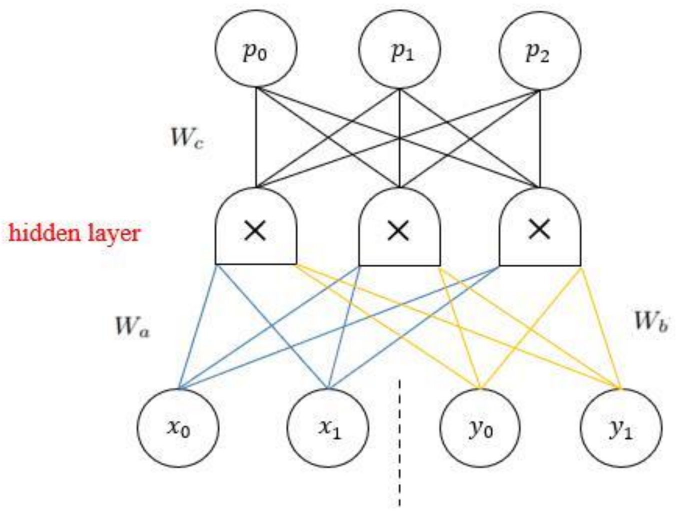

Figure 2: 2-layers SPN

In order to train the weight matrices for Karatsuba, we adjusted our network structure, training dataset and a series of parameters involved. For network, we modified the 2-layer SPN as follows:

The input layer had been set as 2 1*2 matrices, while the output layer had been programmed to give a 1*3 matrix. In addition, the width of the hidden layer that determines the critical number of multiplications of SPN had been pre-determined at 3, as our target was to discover the smaller number of multiplication smaller than the original 4.

As for standardized dataset, we decided to use the SEED parameter as the random seed to split one 4-bit number XY into two 2-bit numbers, X and Y. At the same time, we converted them into matrices as our inputs variables

$$x = x_1 \quad 10^2 + x_0 y = y_1 \quad 10^2 + y_0 \tag{5}$$

On the basis of Karatsuba algorithm,

$$p_0 = x_0y_0p_1 = x_1y_1p_2 = (x_1 + x_0)(y_1 + y_0) - p_1 - p_0 \tag{6}$$

Those were computed to output the matrix [p0 , p1, p2]. We then normalized the input and output matrices to fix into our model. We deployed 100,000 networks as training dataset and 1000 networks as testing dataset for affirmation and the results obtained were as follows.

4. Results

Table 1: The loss values in different seed

Seed values	Greatest Loss	LOSS value
90 - 110	90	0.13
110 - 139	118	0.097
139 - 155	146	0.097
171 - 199	191	0.123
214 - 232	231	0.122
232 - 260	237	0.643
260 - 288	261	0.623
289 - 317	317	0.137
317 - 337	319	0.12
337 - 365	345	0.63

Table 2: Loss value of seed being 146 in different parameter settings

Methods	SEED146
Momentum1= 0.9	0.097796
Momentum2=0.95	0.097795
LearningRateq=0.1	2.078919
LearningRate-ForwardPropgation	2.199175

When the SEED is 146, matrices were shown below and no matrices are exclusive to the projected Karatsuba ones.

$$\begin{bmatrix} 0 & 1.0059828 \\ 1.0059828 & 0 \\ -1.0059828 & 0 \end{bmatrix} \begin{bmatrix} -0.8199405 & 0 \\ 0 & -0.8199405 \\ 0.8199405 & 0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} 0 & 0 & -1.1838703 \\ 0 & 0 & 0 \\ -1.1838703 & -1.1838703 & 0 \end{bmatrix} \tag{8}$$

5. Discussion

These obtained results represent series of attempts done on the prescribed network, however, results were not promising and incomplete due to many limitations that included the shortage of professional knowledge and of other theoretical understandings. In details, both Table 1. and Table 2. demonstrated the relationship between the manipulation of hyperparameters and the Loss value. The latter is an approximate representation of error from the computed to the real value and a variable taken very seriously for in examining the efficacy of our network. In other words, the smaller the loss value is, the more competent is to the network being tested. Specifically, from Table 1. In all tests, even the smallest error value is not ideal, and no target Karatsuba's matrix is generated. In the Table 2., however, it is seen that the adjustment of momentum does not affect the Loss value by any margin, while increase in the learning rate marginally elevates the extent of error. This outcome is a reflection of the first limitation in completing the proposed research.

To be exact, the limitation arising from this problem is the off-range value of the error. In series of trials that have been done, we have tried to manipulate the hyperparameters such as the learning rate and momentum for better training, but the computation does not yield a weight matrix whose error value is as small to be considered as the wanted Karatsuba multiplication matrix. In the Tschannen et al's paper [1], their original network shows error value at about 1000 in failed attempts, as this large value makes the discovery of the correct weights much easier. On the contrary, the proposed network consistently gives out roughly the value of 1, which indicates the difficulty to distinguish each trial. Crucial reflection reveals that such huge difference in the value of error might be attributed to the change of the input, because the wide difference contributes to the value of error so small that it is just intrinsically hard to train the network properly.

As for problems in the aspect of parameters, we found that the Tschannan et al [1] in their paper had initiated a total of 27 trainings in the seed range value of [139, 155], however, only when seed was equal to 146 did the Strassen matrices be discovered. Given this, we suspect with a high confidence that the success of learning in our proposed network largely depends on the selection of the seed value. Consequently, we manipulated a spectrum of seed values to be tested in the module. Besides, we have attempted to increase the size of our testing sets to 50,000 and modified a host of hyperparameters such as learning rate and batch size, but the results remained unpromising to reach a decisive conclusion. Another concern arising necessary attention is the limited scope of output. It is of truth that our entire outputs exclusively relied on the computation of the three p variables as follows.

$$p_0 = x_0y_0p_1 = x_1y_1p_2 = (x_1 + x_0)(y_1 + y_0) - p_1 - p_0 \quad (9)$$

In other words, our network has been learning only the additions and multiplications on these three variables, as opposed to learning the optimization that can reach 3 multiplications for our integer multiplication case. We remain cautious to proposing a possible solution, that is to add another sum layer to serve additional learning after the output layer:

$$xy = p_2 * 10^4 + p_1 * 10^2 + p_0 \quad (10)$$

On the other side, this structural proposal poses intrinsic difficulty for us to implement to see any effective outcomes, since we have been confined on the range of weights [-1, 0, 1] and have not successfully trained our network to discover the Karatsuba matrices. These two factors have insofar limited our additional progress and have indeed displayed the technical and theoretical weaknesses that our group has been experiencing. Until that barrier can be resolved, our knowledge pertaining to the network cannot grow as wished.

Much still remains to be explored about the application of this 2-layer SPN for discovery of Karatsuba algorithms. Proper comparison between other networks in identifying other potential algorithms is also a fascinating area of research in the future that can aid in leading to a more comprehensive outlook of what neural network can be capable of.

6. Conclusion

The paper explored the aspects of neural networks in which the reduction of number of multiplications was most of great interest, due to its applicative potential to save more time in computing. This study then examined the important details of SPN as a potentially candidate to learn the Karatsuba's integer multiplication algorithm. The results concluded that our method led to interesting relationships between hyperparameters and seed values, and led to other interesting questions that we were unable to answer at this point. Although our work did not succeed as previously projected, it did point to the promising direction by which numerous questions have been raised that have not been seen elsewhere. The hypothesis of this research is unfortunately neither accepted nor refuted. Instead, the work laid a solid foundation on which more and possibly better implementation can be done for ideal improvement.

References

[1] M. Tschannen, A. Khanna, and A. Anandkumar, "StrassenNets: Deep Learning with a Multiplication Budget," in International Conference on Machine Learning, Jul. 2018, pp. 4985–4994.

[2] E. Nurvitadhi et al., "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, New York, NY, USA, Feb. 2017, pp. 5–14, doi: 10.1145/3020078.3021740.

[3] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An Architecture for Ultra-Low Power Binary-Weight CNN Acceleration," ArXiv160605487 Cs, Feb. 2017. Retrieved from: http://arxiv.org/abs/ 1606. 05487.

[4] V. Elser, "A Network That Learns Strassen Multiplication," J. Mach. Learn. Res., 2016, vol. 17, pp. 1–13.

[5] T. Adel, D. Balduzzi, and A. Ghodsi, "Learning the Structure of Sum-Product Networks via an SVD-based Algorithm," in Proceeding of the Thirty-First Conference on Uncertainty in Artificial Intelligence, Jul. 2015, pp.32-41.

[6] A. L. Friesen and P. Domingos, "The Sum-Product Theorem: A Foundation for Learning Tractable Models," in Proceedings of the 33rd International Conference on Machine Learning, Nov.2016, pp.1909-1018.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in Proceedings of the 26th Annual International Conference on Machine Learning, New York, NY, USA: Association for Computing Machinery, 2009, pp. 609–616.

[8] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," in 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Nov. 2011, pp. 689–690, doi: 10.1109/ ICCVW. 2011.6130310.

[9] S. Mishra and M. Pradhan, Implementation of Karatsuba Algorithm Using Polynomial Multiplication, Indian Journal of Computer science and engineering, Feb. 2012, vol. 3, No.1.

[10] A. Weimerskirch and C. Paar, Generalizations of the Karatsuba Algorithm for Polynomial Multiplication. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.4028.