Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 9, Issue, 06, pp.51823-51826, June, 2017

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

EFFECTIVE FORWARD AND BACKWARD PROPAGATION METHOD FOR PIXEL-WISE CONVOLUTIONAL NEURAL NETWORK

*,1Bikramjit Chowdhury, 1Dr. Meena, K., 2Ruchika Vyas and 2Aditya Kumar Sinha

1Department of Computer Science and Engineering, Veltech Dr.RR & Dr.SR University, Chennai, Tamil Nadu
2Centre for Development of Advanced Computing (C-DAC), Pune, India

ARTICLE INFO

Article History:
Received 24th March, 2017
Received in revised form
08th April, 2017
Accepted 10th May, 2017
Published online 20th June, 2017

Key words:

Artificial Neural Network,
Convolutional Neural Network,
Fully connected convolution neural
Network, OpenMP, MPI.

ABSTRACT

Satellite Image is used to analysis the ground area to identify the object. The main barrier of this image processing method is the pixel-wise analysis. The fully convolutional neural network is a popular method to process this type of image processing. However, train stage is taken the maximum computation of this method. Parallelize version of this method to reduce computation time is researching topics, now. But little work is done on optimize the complexity of the algorithm of this method. In this paper, we find an optimized algorithm solution of this algorithm. Also, the result of the parallel version of the proposed algorithm is discussed.

Citation: Rachana, A. and Raj Kumar, B., 2017. "Effective forward and backward propagation method for pixel-wise convolutional neural network", *International Journal of Current Research*, 9, (06), 51823-51826.

INTRODUCTION

The Convolutional Neural Network is the extracted version of the artificial neural network, which was first introduced on 1943 based on cat vision. The Artificial Neural Net is the artificial improvement of biological Neural functionality. Like, biological neuron it takes an input value and process it then passes it, by applying Linear operation, nonlinear operation and then sends to the next layer neuron(hidden layer).Like previous neuron the next neuron process (linear-nonlinear operation) the input data and trance to the next neuron. This process is continued until it reaches the output layer, where the score is calculated and compared it with expected score to calculate the error. Base on the error weights of each neuron is updated to reduce it. The Convolutional Neural Network first introduced on 1997 by LeNet-5[1]. It becomes more popular by Alex Net[2] in 2012 when a parallel version by GPU and addition future like Dropout concept was introduced. They introduced their work by a library like cudaCNN [2]. As per LeNet-5[1] the basic convolutional neural network containing convolutional layer, pooling layer, fully connected layer.

Corresponding author: **Bikramjit Chowdhury,**
Department of Computer Science and Engineering, Veltech Dr. RR & Dr. SR University, Chennai, Tamil Nadu.

The convolutional layer performs the same operation (linear-non-linear) like the neuron on the artificial neuron network. In the pooling layer, the future is reduced. It helps to normalize the learning method. The fully connected layer is like the output layer of the Artificial Neural Network; here error and score are calculated. The main barrier in the convolutional neural network is that input pixel is merged with other pixel value in the convolutional layer. So, it could not fit for a problem where pixel-wise leveling is required. The fully connected convolutional neural network [3, 4] solves this problem by mapping individual pixel(input) with output level. Here input size is same as output size in each layer. In this paper, we proposed an optimal algorithm of this paper[5], which used a similar method for pixel-wise mapping in the fully connected convolutional neural network[3,4], which gives same expected result with less complexity. This algorithm is applied to the satellite data set (D-STL). It contains RGB band and multi-multi-spectrum band image(IR) data set. All these band image files are merged into a single image. The edging and reduce the pixel value between 0 to 1.0 is done before merging the image. The parallel version of this algorithm is implemented on OpenMP API, which facilitated program by dividing it into separate work share model. Here parent thread initiates child threads in a parallel region after execution of the parallel region child threads merged back to the parent thread.

This algorithm is also used in MPI cluster with OpenMP. MPI is massage passing interface which is popularly used in HPC cluster. Here we used 2 processes in a single machine. In this project, Intel 64 bit hardware with i5 processor and 3.3GB RAM is used for providing computation power.

Related work

Parallelization in Artificial Neural Network is an old approach. As previously mentioned papers [2, 3, 4] optimizes the neural network by GPU cluster. In other are of research work is done on the base of OpenMP, MPI in a computer cluster [5,6]. However, research on optimizing the complexity in the neural network is very less in number. We, here to improve the performance of the algorithm not only by the developing parallel version of the algorithm but also reduced the complexity of the algorithm.

Region-wise analysis of pixel-wise convolutional neural network

Region 1	Region 2	Region3
Region4	Region 8	Region5
Region6	Region 9	Region7

Figure 1. Region wish computation in convolutional layer

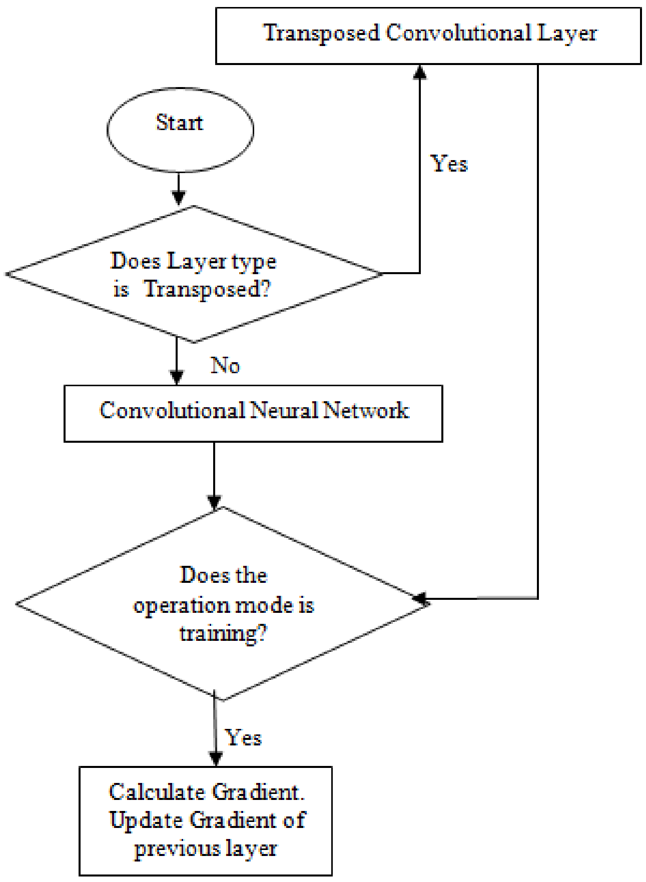
Transposed Convolutional Layer
Start
Yes
Does Layer type is Transposed?
No
Convolutional Neural Network
Does the operation mode is training?
Yes
Calculate Gradient. Update Gradient of previous layer

Figure 2. Program flow

Here, the linear and nonlinear operation is performed on a single and converted into a single output value. For$N \times M$ input image$(N \quad wr) \times (M \quad wr)$output image is create for a single future of a layer. If the layer containing F future then total$F \times (N \quad wr) \times (M \quad wr)$ output is crteated. It's complexity very similar to the common convolutional neural network. For reduced computation gradient is calculated in forwarding operation is stored in local variables (Fig2.).

Optimize convolutional neural network by region wise pixel selection

In this method(Fig1.), Input is divided into a number of regions, and then the result is calculated.

Region 1: rows and columns of pixel (input value) in this region is less than equal to wr, wc

Region 2: Columns of the pixel (input value) in this region is greater than wc, but less then $M \quad wc$ and row is less then equal to wr.

Region 3: Columns of the pixel (input value) in this region is greater than $M \quad wc$ and row is less thenwr.

Region 4: Columns of the pixel (input value) is less than wc and row value is greater then wr but less then equal to $N \quad wr$.

Region 8: Column value of pixel (input value) is less than $M \quad wc$ and row valuye is greater then wr but less then $N \quad wr$.

Region 5: Column value of the pixel in this area is larger then $M \quad wc$ and equal to M and row value is greater then wr but less then $N \quad wr$.

Region 6: Column value of the pixel in this area is less than, equal to wc and row value is greater then $N \quad wr$ but less then, equal to N.

Region 9: Column value of the pixel in this area is larger then wc and equal to $M \quad wc$ and row value is greater then $N \quad wr$ but less then, equal to N.

Region 7: Column value of the pixel in this area is larger then wc and equal to $M \quad wc$ and row value is greater then $N \quad wr$ but less then, equal to N.

The operation of region wise pixel selection

- Here multiplication operation is done in every pixel individually.
- If pixel (input value) is in the Region 1 then every weight which rows and columns are less than equal to is added and the sum of this value is multiplied by the pixel(input value).
- If pixel (input value) is in the Region 2 then the addition of every row wish addition every weight which has row value is less than, equal to the row value of the pixel (input value) is multiply by the pixel (input value).
- For Region 3 pixel (input value) is multiplied with the summation of weights which have larger row and column value than the pixel (input value).
- A pixel in the Region 4 is multiplied by the sum of column sum of weights which have less column value then this pixel (input value).
- A pixel in the Region 5 is multiplied by the sum of column sum of weights which have greater column value then this pixel (input value).

- A pixel in the Region 8 is multiplied by the sum of column sum of all weights.
- A pixel in the Region 6 is multiplied by the sum of weights, which have greater row and column value.
- A pixel in the Region 9 is multiplied by the sum of the sum of rows of weights, which have greater row.
- A pixel in the Region 7 is multiplied by the sum of weights, which have greater row and column value.

The complexity of region wise pixel selection

- For reduce operation complexity row wish, column wish sum and the total sum of the weights are previously computed for each forward operation.
- Calculate the total of the weights from weight matrix take $O(wr \times wc)$ operation.
- Calculate total of the column wish weights take $O(WR)$ for each column so total take $O(wr \times wc)$
- Similarly, row wish sum is calculated in $O(wr \times wc)$ operation.
- This loop can merge to a single operation which takes$O(wr \times wc)$.
- Let say the pixel (input value) position is (x,y).
- For Region 1 operation complexity is $O(x \times y)$ for all elements, total operation $O(wr^2 \times \ wc^2)$
- For Region 2 operation complexity is $O(x)$, for all element $O((M \quad 2wc) \times wr)$
- For Region 3 operation complexity is $O((N \quad x) \times y)$ for all elements $O(wr^2 \times \ wc^2)$.

Table I. **Serial and OpenMP Code performance**

Size of Data Set	Serial Code	Parallel version (OpenMP) of Optimize Code	
		4 Threads	8 Threads
644MB	30 minute	12 minute	3.6 minute
1 GB	1 hour	24 minute	7. 2 minute
1.5 GB	1hour 30 minute	36 minute	10.8 minute

Table II. **Serial and OpenMP-MPI Code performance**

Size of Data Set	Serial Code	Parallel version (OpenMP) of Optimize Code (6 process)	
		4 threads	8 threads
644MB	30minute	14 minute	5 minute
1 GB	1 hour	30 minute	15 minute
1.5GB	1hour 30 minute	40 minute	20 minute

- For Region 4 operation complexity is $O(y)$for all elements $O((N \quad 2wr) \times \ wc)$
- For Region 5 operation complexity is $O(M \quad y)$ for all elements$O((N \quad 2wr) \times \ wc)$
- For Region6 operation complexity is $O\big((N \quad x) \times y\big)$, for all elements $O(wr^2 \times \ wc^2)$
- For Region 8 operation complexity is $O(1)$, for all elements $O((N \quad 2wr) \times (M \quad 2wc))$
- For Region9 operation complexity is $O((N \quad x))$, for all elements $O((M \quad 2wc) \times wr)$
- For Region 7 operation complexity is $O((N \quad x) \times (M \quad y))$, for all elements $O(wr^2 \times \ wc^2)$

Total operation perform for a single convolutional forward operation by this method is $O(4 \times (wr \times wc)^2 + 2 \times (M \quad 2wc) \times wr) + 2 \times (N \quad 2wr) \times \ wc + (N \quad 2wr) \times (M \quad 2wc))$. When $wr = wc$, $N = M$then $O(4 \times wr^4 + 4 \times (N \quad 2wr) \times wr) + (N \quad 2wr)^2)$.

When $wr \ll N$, then $O((N \quad 2wr)^2)$

Total operation perform for a single convolutional layer in LeNet is $O((N^2 \times wr^2)^2)$. When $N = M, and\ wr = wc$ If $wr \ll N$ then $O(N^4)$.

The parallel implementation is done by OpenMP and MPI by following parts.

OpenMP: Work sharing is applying on each layer. The gradient is calculated for each forward operation. In forwarding operation of each convolutional layer gradient of the previous layer is calculated.

MPI: In the model, Image is dividing and shared by Master to slave process, the beginning and ending are given to each process.

Gradient value for each forward operation is calculated and shared with other processes in forwarding operation. After training and testing result of percentage of each is send back to Master process.

Result analysis

Form Table I and Table II it is clear that this model gives better performance with 5 processes.

Conclusion

More optimal algorithm for pixel-wise analysis is introduced. A parallel version of the method is introduced here. Also, the resulted performance is analyzed. As Satellite object analysis is complicated, the more effective algorithm is needed to analyze satellite images. So, further study has to be done in this field.

Acknowledgement

I would like to thank Dr. K. Meena. of Veltech University Chennai for her valuable suggestions and Mr. Aditya Kumar Sinha and Miss. Ruchika from C-DAC Pune for initiating research in the university providing platform to students with the opportunity to gain knowledge about the area of research. This work is done as part of M.Tech dissertation work.

REFERENCES

Chen, L.C. Papandreou, G. Kokkinos, I. Murphy, K. and Yuille, A. L. 2015. “Semantic image segmentation with deep convolutional nets and fully connected CRFs”. In *ICLR*, 2015.

Daniel del and Hoyo Rodriguez, “Optimization of Backpropagation Learning Algorithm on MLP Networks”, 2 Nov 2012.

Hyeonwoo Noh, Seunghoon Hong and Bohyung Han, Department of Computer Science and Engineering, Postech Korea, “Learning Deconvolution Network for Semantic Segmentation”, arXiv:1505.04366v1 [cs.CV] 17 May 2015

Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, in NIPS, 2012.

LeCun, Y. Bottou, L. Bengio, Y. and Haffner, P. 1998. “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, 1998.

Long, J. Shelhamer, E. and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015

Schuessler, O. and Loyola, D. 2011. ”Parallel Training of Artificial Neural Networks Using Multithreaded and Multicore CPUs”. In: Dobnikar A., Lotrič U., Šter B. (eds) Adaptive and Natural Computing Algorithms. ICANNGA 2011. Lecture Notes in Computer Science, vol 6593. Springer, Berlin, Heidelberg.

****