Comparative evaluation of performance and scalability of convolutional neural network implementations on a multisystem HPC architecture

Sunil Pandey¹, Naresh Kumar Nagwani², Shrish Verma³

¹Research Scholar, Dept of Computer Science and Engineering, NIT Raipur, India
²Associate Professor, Dept of Computer Science and Engineering, NIT Raipur, India
³Professor, Dept of Electronics and Communications Engineering, NIT Raipur, India

sys_admin@nitrr.ac.in

Abstract. The convolutional neural network training algorithm has been implemented for a central processing unit based high performance multisystem architecture machine. The multisystem or the multicomputer is a parallel machine model which is essentially an abstraction of distributed memory parallel machines. In actual practice, this model corresponds to high performance computing clusters. The proposed implementation of the convolutional neural network training algorithm is based on modeling the convolutional neural network as a computational pipeline. The various functions or tasks of the convolutional neural network pipeline have been mapped onto the multiple nodes of a central processing unit based high performance computing cluster for task parallelism. The pipeline implementation provides a first level performance gain through pipeline parallelism. Further performance gains are obtained by distributing the convolutional neural network training onto the different nodes of the compute cluster. The two gains are multiplicative. In this work, the authors have carried out a comparative evaluation of the computational performance and scalability of this pipeline implementation of the convolutional neural network training with a distributed neural network software program which is based on conventional multi-model training and makes use of a centralized server. The dataset considered for this work is the North Eastern University’s hot rolled steel strip surface defects imaging dataset. In both the cases, the convolutional neural networks have been trained to classify the different defects on hot rolled steel strips on the basis of the input image. One hundred images corresponding to each class of defects have been used for the training in order to keep the training times manageable. The hyperparameters of both the convolutional neural networks were kept identical and the programs were run on the same computational cluster to enable fair comparison. Both the convolutional neural network implementations have been observed to train to nearly 80% training accuracy in 200 epochs. In effect, therefore, the comparison is on the time taken to complete the training epochs.
1. Introduction
Convolutional neural networks refer to an important class of deep learning artificial intelligence algorithms which have distinguished themselves through their demonstrably superior performance in a multitude of computer vision tasks like object detection, pattern recognition, image classification, scene understanding, etc. of numerous scientific, engineering and technological fields. In the last decade, convolutional neural networks have been extensively applied over a range of tasks in several domains [1-20]. Superior classification performance and accuracies have been obtained through the application of convolutional neural networks in classification problems in comparison to other classification algorithms for the same task [21-24]. A very good introduction to deep learning algorithms and convolutional neural networks is available in [25]. Despite these successes of convolutional neural networks, there is an emerging consensus among several researchers and practitioners of deep learning that the adaptation of convolutional neural networks for real-world imaging applications is facing a roadblock largely due to their fairly long training times which are known to extend from weeks to even few months on computational systems with advanced computer architectures [26-28]. In light of the same, multifaceted research and development activities are being undertaken in different directions with the objective of bringing down the training times of deep convolutional neural networks [29-44].

Pipelined computation is an established methodology for speeding up computations. In this methodology, a large computational task is broken down into several smaller sequential sub-tasks such that the output of one sub-task becomes the input of another. Pipeline parallel programming technique and the composing of computations as high-level parallel computations has been explained in [45]. On the fly pipeline parallelism is brought out in [46]. The construction of parallel pipelines is the subject matter of [47]. Reasons for how pipelines can fail are explained in [48]. Pipeline scheduling and the analytical modelling of computational pipelines are explained in [49] and [50] respectively. The requirement for the design and development of parallel and scalable deep learning algorithms for high performance computing architectures has been clearly elucidated by the authors in [51]. The paper discussed the requirement of high performance computing for accelerating the computation and time intensive training of convolutional neural networks. In the paper, a multi-core implementation of the convolution operation was carried out. A pipeline design of the convolutional neural network for implementation on high performance computing architectures has been proposed by the authors in [52]. Analysis of the convolutional neural network feature extraction pipeline on the surface defects dataset indicates that it is possible to obtain performance gains of up to 18X by combining the performance gains obtained through pipeline parallelism with task mapping and parallel execution of the pipeline blocks on the nodes of a multisystem, i.e., a computational cluster. Some aspects considered important for the programming of convolutional neural networks for implementation on multisystem high performance computing architectures have been discussed by the authors in [53]. In the current paper, that work has been extended to performance and scalability evaluation of the high performance deep learning convolutional type neural network on a metal surface defects classification problem. An implementation of pipeline based convolutional neural network design with task mapping on different numbers of nodes i.e., central processing units of a multisystem high performance computing architecture has been made and adapted to this problem. A comparative evaluation of the performance and scalability of this implementation has been done with a freely obtainable distributed neural network software program which is based on conventional multi-model training and makes use of a centralized server for aggregation and distribution of the model parameters.

2. Image Dataset Description
The Northeastern University’s surface defect database [54] is a collection of six types of archetypal surface defects of hot-rolled steel strips. The defects which have been considered include the following: Rolled-in Scale (RS), Patches (Pa), Crazing (Cr), Pitted Surface (PS), Inclusions (In) and Scratches (Sc). The image dataset is a collection of 1,800 grayscale images comprising of 300 samples
each of the six different types of archetypal hot-rolled steel surface defects. The sample images of the six different types of archetypal hot-rolled steel surface defects are shown in Figure 1 below which has been reproduced from the link in [54]. The original resolution of all the images is 200 pixels × 200 pixels.

It can be observed that within each class of defects, there exists large differences in visual appearance. Thus, for example, the first and fourth images corresponding to the pitted surface defect have rather different visual appearance. At the same time, some of the defects across the classes have similar aspects, e.g., to the untrained human observer, it is easy to confuse some of the images of rolled-in scale with crazing. The problem is worsened due to the differences in material and illumination, and due to this, the grayscale images within a certain defect class exhibit substantial variance. The Northeastern University’s surface defect dataset poses two difficult challenges, which are the samples from the same defects class exhibiting considerable differences in appearance while those falling in different classes having similar aspects. Also, the images appear differently due to differences in material and illumination.

![Figure 1: Six Different Types of Archetypal Hot-Rolled Steel Surface Defects [4]](image)

3. **Convolutional Neural Network Pipeline Design**

The top level convolutional neural network pipeline design for the hot-rolled steel strip surface defects classification problem is depicted in Figure 2. A total of six independent convolutional neural network pipelines can be observed in the pipeline design of Figure 2. Each of the pipeline considered has a loader, four “cnp” stacks where “cnp” stands for convoluter, nonlinear mapper and pooler respectively. The convolutional neural network also has four sets of filter banks comprising of twenty five numbers of convolution kernels of dimensions 11 x 11 each. There is a flattening block used for
conversion from two dimensional matrix representation to one dimensional feature vector for use by the classifier. The classifier shown is internally constituted of one classification unit for each pipeline. Input images have been uniformly distributed to the image loaders to the extent practicable. One hundred images corresponding to each class in the dataset have been included for training the convolutional neural network based classifier.

Figure 2: CNN Pipeline Design Comprising of Six Pipelines

Figure 3 depicts another top-level convolutional neural network pipeline design for the same hot-rolled steel strip surface defects classification problem. However, as can be noted in this design, the six pipelines have been replaced by twelve pipelines. The remaining details of the convolutional neural network structure and design are the same as in the case of the twelve pipeline convolutional neural network.

Figure 3: CNN Pipeline Design Comprising of Twelve Pipelines
4. Multisystem Architecture

The multisystem high performance computing architecture comprises of several Von Neumann nodes which are connected to and communicate with each other through a communication interconnect. This is also referred to as the distributed parallel architecture. It is not possible for a process running on any node to directly access the memory of another node in this architecture. Therefore, this is also known as the No Remote Memory Access Architecture. As can be seen from Figure 4, each processor CPU is connected to exclusive local memory, i.e., no other central processing unit has direct access to it. Each node comprises of a network interface, that mediates the connection to a communication network or network interconnect. A serial process runs on each central processing unit that can communicate with other processes on other central processing units by means of the network. A problem is solved cooperatively by sending messages back and forth between the processes. The interconnect is generally Ethernet, Myrinet, Infiniband, but can, in principle, also be the World Wide Web.

![Multisystem Parallel Computer Model](image)

**Figure 4:** Multisystem Parallel Computer Model

5. Parallel Programming Model

The tasks and channels model has been used for programming multicomputer architectures. The model comprises of one or more tasks which can execute simultaneously as depicted in Figure 5. The tasks comprising a parallel computation can differ in number when a program is executing.

![Task and Channel Parallel Computation Model](image)

**Figure 5:** Task and Channel Parallel Computation Model

A task is a sequential program, local memory with a set of inports and outports that define its interfaces. Tasks can perform the following actions besides reads and writes to their local memory: send messages on their outports, receive messages on their inports, create new tasks, and terminate. Send operations are asynchronous operation in that they complete and return the control immediately. However, the receive operation is synchronous and causes the task execution to block until a message becomes available. Outport and inport pairs are connected through message queues called channels. Channels can be created and deleted, messages contain references to channels with port information,
therefore the connectivity is dynamic during the program execution. Mapping of tasks to physical processors is possible in a numerous ways. The choice of a particular mapping does not have any effect on the program semantics. It is also possible for multiple tasks to be mapped to a single physical processor.

6. Metrics for Parallel Programs

6.1 Speedup
Speedup is a metric which is an indicator of the computational performance of a parallel program. Speedup is given by the expression

\[
speedup = \frac{\text{wall-clock time of serial execution}}{\text{wall-clock time of parallel execution}} = \frac{1}{S+N}
\]

where \( S \) is the scalar fraction of the code,
\( P \) is the parallel fraction
\( N \) is the number of CPUs or cores.
By normalization, \( S + P = 1 \)

Ideally, the speedup of a Parallel Program should be \( N \)

6.2 Scalability
Scalability or efficiency of a parallel program is defined as the ratio of the speedup and the number of CPUs.

\[
\text{Scalability} = \frac{\text{Speedup}}{\text{Number of CPUs}} = \frac{\text{Speedup}}{N}
\]

Efficiency is a measure of how well a parallel program scales when the number of CPUs available for computation are increased. Ideally, the efficiency of a parallel program should be 1 (100%)

Figure 6 shows the parallel speedup with the number of processors and the plot of what is considered typical success in parallelization.

![Figure 6: Speedup vs Number of CPUs](image)
7. Results and Discussion

The computational results are provided for the six pipeline convolutional neural network of Figure 3. The plot of the cost functions, (or the mean square error values) vs the number of training epochs is shown in Figure 7. It can be seen from the plot that the mean square error values of all the pipelines start to drop relatively faster in the initial epochs. However, as the epochs increase, the convergence is relatively slower. The mean square error values of the six pipelines start to converge to values close to 0.20 around the 150th epoch.

![Figure 7: Cost Function versus Epochs / Iterations](image1)

The plot of the training set accuracy vs the number of iterations for the proposed pipeline parallel implementation of the convolutional neural network can be seen from Figure 8. It can be observed from this plot that the training set accuracies increase steadily with increasing number of epochs and start saturating after 125 iterations or so. From this point on the training accuracies increase relatively gradually and approach close to 80% at the 200th epoch. A similar trend has been observed with twelve pipeline convolutional neural network architecture. The accuracy vs epochs varies similarly in the case of the distributed neural network program also.

![Figure 8: Training Set Accuracy versus Number of Epochs / Iterations](image2)

Table 1 shows the computation times and speedups of both the programs. The pipelined CNN run on one, two and six compute nodes was comprised of six independent pipelines.
while the CNN on four and twelve nodes or physical processors comprised twelve independent pipelines.

Table 1: Computation Times and Speedups of Proposed Pipelined CNN and the Distributed NN

| Nodes | Training Time(s) | Speedup | Training Time (s) | Speedup |
|-------|------------------|---------|-------------------|---------|
|       | Dist NN          |         | Pipelined CNN     |         |
| 1     | 46000            | 0.83    | 38252             | 1.0     |
| 2     | 25572            | 1.50    | 21184             | 1.81    |
| 4     | 13727            | 2.79    | 12200             | 3.14    |
| 6     | 8251             | 4.63    | 6808              | 5.62    |
| 8     | 7024             | 5.44    | *                 | *       |
| 12    | 4172             | 9.17    | 3595              | 10.64   |
| 16    | 3673             | 10.41   | *                 | *       |

*Pipelined CNN not run on these node numbers since they are not divisors/multiples of 6 or 12

The convolutional neural networks have been trained to classify the different defects on the basis of the input image. One hundred images corresponding to each defect were used for the training in order to keep the training times reasonable. The hyperparameters of both the convolutional neural networks were kept identical and both the programs were run on the same multisystem with each node having 8 core AMD Opteron processors and 8GB RAM. Both the CNNs trained to nearly 80% training accuracy in 200 epochs. In effect, therefore, the comparison is on the time taken to complete the epochs.

The plot of Figure 9 shows the speedup vs number of nodes for ideal scalability, scalability of conventional distributed neural network and the pipelined distributed neural network. The speedup vs number of processors can be seen to follow the trend of “typical success” of Figure 6. However, as can be seen from the plot in the proposed pipeline parallel implementation of the convolutional neural network is more scalable than the conventional distributed neural network.

![Figure 9: Performance and Scalability on the NEU-CLS Dataset](image)

Some of the prior accuracy results on the Northeastern University (NEU) surface defect database are summarized as follows. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects classification has been proposed in [55] while in [56], a pyramid feature fusion and global context attention network for automated surface defect detection. Both
methods in [55] and [56] have reported classification accuracies in the range of 80% which is comparable to the convolutional neural network implementations of this paper. There are important differences however in that the deep learning convolutional neural network implementations make use of automated feature extraction whereas the two methods highlighted above make use of hand-crafted features for the specialized task. Further, the convolutional neural network training has been done on a subset of the data only whereas the two other methods have made use of the entire dataset. In a third method [57], which is based on the fusion of multiple hierarchical features, accuracies of around 92% have been reported. However, this is again based on supervised learning through hand-crafted features specialized to the particular task at hand.

8. Conclusions

A high performance deep learning convolutional type neural network algorithm for the metal surface defects classification problem has been designed as two computational pipelines with six and twelve numbers of independent pipelines each. It has been implemented using the task parallel model on a central processing unit based multisystem high performance computing architecture, i.e., a high performance computing cluster. It can be seen that the proposed pipeline parallel implementation of the convolutional neural network outperformed the distributed conventional convolutional neural network. When the number of independent pipelines in a convolutional neural network model are mapped to the same number of nodes of the multisystem, it can be seen that the speedup is 5.62 for six nodes and 10.64 for twelve nodes. The distributed conventional convolutional neural network is not able to attain these speedups even with eight and sixteen nodes respectively. It is also observed that the proposed pipeline parallel implementation of the convolutional neural network is more scalable than the conventional distributed convolutional neural network program. The proposed pipeline parallel implementation of the convolutional neural network has both higher speedup and greater scalability when compared with the conventional distributed convolutional neural network program.

References

[1] Zezhou Cheng Qingxiong Yang and Bin Sheng Deep Colorization IEEE International Conference on Computer Vision (ICCV) Santiago 2015 pp. 415-423.
[2] Zhang R. Isola P. and Efros A.A. Colorful Image Colorization European Conference on Computer Vision 2016 - Springer vol. 9907 2016.
[3] Larsson G. Maire M. and Shakhnarovich G. Learning Representations for Automatic Colorization Computer Vision( ECCV) Lecture Notes in Computer Science - Springer vol. 9908 2016.
[4] Hwang Jeff and You Zhou Image Colorization with Deep Convolutional Neural Networks Stanford University 2016.
[5] Andrew Owens et al. Visually Indicated Sounds in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Las Vegas 2016 pp. 2405 - 2413.
[6] I Sutskever O. Vinyals and Q.V. Le Sequence to Sequence Learning in Proc. Advances in Neural Information Processing Systems 27 2014 pp. 3104–3112.
[7] K. Cho et. al. Learning phrase representations using RNN encoder-decoder for statistical machine translation in Proc. Conference on Empirical Methods in Natural Language Processing 2014 pp. 1724–1734.
[8] Zhang Jiajun and Zong Chengqing Deep Neural Networks in Machine Translation: An Overview IEEE Intelligent Systems vol. 30 no. 5 pp. 16-25 Sept-Oct 2015.
[9] Krizhevsky I. Sutskever and G. Hinton ImageNet classification with deep convolutional neural networks in *NIPS Proceedings* 2012.

[10] G. Howard Some improvements on deep convolutional neural network based image classification in *International Conference on Learning Representation (ICLR)* Banff Canada 2014.

[11] D. Erhan C. Szegedy A. Toshev and D. Anguelov Scalable Object Detection Using Deep Neural Networks in *IEEE Conference on Computer Vision and Pattern Recognition Columbus* 2014 pp. 2155-2162.

[12] D. Erhan C. Szegedy and A. Toshev Scalable object detection using deep neural networks in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2014.

[13] Alex Graves Generating Sequences With Recurrent Neural Networks 2014. [Online]. https://arxiv.org/pdf/1308.0850.pdf

[14] Ilya Sutskever James Martens and Geoffrey E Hinton Generating text with recurrent neural networks in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* New York NY 2011 pp. 1017-1024.

[15] Andrej Karpathy and Li Fei-Fei Deep Visual-Semantic Alignments for Generating Image Descriptions in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Boston Massachusetts 2015 pp. 3128 - 3137.

[16] Ayushi Chahal Preeti Gulia Deep Learning: A Predictive IoT Data Analytics Method *International Journal of Engineering Trends and Technology* vol. 68 no. 7 2020.

[17] P. Seetha Subha Priya S. Nandhinidevi M. Thangamani S. Nallusamy A Review on Exploring the Deep Learning Concepts and Applications for Medical Diagnosis *International Journal of Engineering Trends and Technology* vol. 68 no. 10 2020.

[18] Sangeeta Preeti Gulia Deep learning based combating strategy for COVID-19 induced increased video consumption *International Journal of Engineering Trends and Technology* vol. 68 no. 7 2020.

[19] Ferdinand Kartriku Robert Sowah Charles Saah Deep Neural Network: An Efficient and Optimized Machine Learning Paradigm for Reducing Genome Sequencing Error *International Journal of Engineering Trends and Technology* vol. 68 no. 9 2020.

[20] Ramya T.E. Marikkannan M. Investigations on Combinational Approach for Processing Remote Sensing Images Using Deep Learning Techniques *International Journal of Engineering Trends and Technology* vol. 67 no. 8 2019.

[21] Ciresan Dan Ueli Meier and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition* 2012 pp. 3642-3649.

[22] Ciresan Dan Ueli Meier Jonathan Masci Luca M. Gambardella and Jurgen Schmidhuber. 2011. Flexible High Performance Convolutional Neural Networks for Image Classification. in *International Joint Conference on Artificial Intelligence* 2013 pp. 1237–1242.

[23] Lawrence Steve C. Lee Giles Ah Chung Tsoi and Andrew D. Back. Face Recognition: A Convolutional Neural Network Approach. *IEEE Transactions on Neural Networks* 1997 Volume 8; Issue 1 pp. 98-113

[24] Russakovsky O. Deng J. Su H. et al. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis* vol. 115 2015
[25] Yann LeCun Yoshua Bengio and Geoffrey Hinton Deep Learning Nature vol. 521 p. 436 2015.
[26] X. W. Chen and X. Lin Big Data Deep Learning: Challenges and Perspectives IEEE Access vol. 2 pp. 514-525 2014.
[27] M.M. Najafabadi et. al. Deep learning applications and challenges in big data analytics Journal of Big Data p. 1 2015.
[28] P. Angelov and A. Sperduti Challenges in Deep Learning in European Symposium on Artificial Neural Networks Computational Intelligence and Machine Learning Bruges (Belgium) 2016 pp. 27-29.
[29] Xian-He Sun Yong Chen and Surendra Byna Scalable Computing in the Multicore Era in Proceedings of the Inaugural Symposium on Parallel Algorithms Architectures and Programming Hefei: University of Science and Technology of China Press 2008.
[30] M. Tanveer M.A. Iqbal and F. Azam Using Symmetric Multiprocessor Architectures for High Performance Computing Environments International Journal of Computer Applications vol. 27 no. 9 2011.
[31] M. B. Giles and I. Reguly Trends in high-performance computing for engineering calculations in Philosophical Transactions of the Royal Society of America 2014.
[32] J. Dean et. al. Large Scale Distributed Deep Networks in 25th International Conference on Neural Information Processing Systems Lake Tahoe Nevada 2012 pp. 1223-1231.
[33] J. Hauswald et. al. DjiNN and Tonic: DNN as a service and its implications for future warehouse scale computers in Proceedings 42nd Annual International Symposium on Computer Architecture (ISCA) Portland OR USA 2015 pp. 27-40.
[34] V. Hegde and S. Usmani. (2016) Parallel and Distributed Deep Learning Stanford University Online Report.
[35] M. Bouache and J. Glover Deep Learning GPU-Based Hardware Platform Hardware and Software Criteria and Selection in ICS-2016 Istanbul Turkey 2016.
[36] Q. Le et. al. On Optimization Methods for Deep Learning in Proceedings of the International Conference on Machine Learning Washington 2011.
[37] J. Keuper and F.J. Pfreundt Asynchronous Parallel Stochastic Gradient Descent A Numeric Core for Scalable Distributed Machine Learning Algorithms in Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments Austin TX USA 2015.
[38] V. Vanhoucke A. Senior and M. Z. Mao Improving the speed of neural networks on CPUs in Proceedings of the Deep Learning and Unsupervised Feature Learning NIPS Workshop Granada Spain 2011.
[39] S. Gupta A. Agrawal K. Gopalakrishnan and P. Narayanan Deep Learning with Limited Numerical Precision Journal of Machine Learning Research 37 2015.
[40] S Chetlur et. al. (2014) cuDNN: Efficient Primitives for Deep Learning. [Online]. https://arxiv.org/pdf/1410.0759.pdf
[41] Delong. Practical Guide to Matrix Calculus for Deep Learninghttp://www.psi.toronto.edu/~andrew/papers/matrix_calculus_for_learning.pdf [Online]
[42] Baoyuan Liu Min Wang H. Foroosh M. Tappen and M. Penksy Sparse Convolutional Neural Networks in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Boston MA 2015 pp. 806-814.
[43] Ionescu O. Vantzos and C. Sminchisescu Matrix Backpropagation for Deep Networks with Structured Layers in IEEE International Conference on Computer Vision (ICCV) Santiago 2015 pp. 2965-2973.

[44] Y. Zhang and S. Zhang Optimized Deep Learning Architectures with Fast Matrix Operation Kernels on Parallel Platform in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence Herndon VA 2013 pp. 71-78.

[45] Mario Rossainz-López Manuel I. Capel Odon D. Carrasco-Limón Fernando Hernández-Polo Bárbara E. Sánchez-Rinza Implementation of the Pipeline Parallel Programming Technique as an HLPC: Usage Usefulness and Performance Annals of Multicore and GPU Programming 4 (1). ISSN: 2341-3158.

[46] I-Ting Angelina Lee Charles E. Leiserson Tao B. Scharl Jim Sukha Zhuping Zhang On-the-Fly Pipeline Parallelism ACM Transactions on Parallel Computing 2 No. 3

[47] Halling-Brown M Shepherd AJ. Constructing computational pipelines. Methods Mol Biol. 2008;453:451-70.

[48] Vivien Marx When Computational Pipelines Go Clank. Nature Methods 17 659–662 (2020)

[49] Saurav Chatterjee and Jay Strosnider Distributed Pipeline Scheduling: A Framework for Distributed Heterogeneous Real-Time System Design The Computer Journal 38 No. 4 1995

[50] Navarro R. Asenjo S. Tabik and C. Cascaval Analytical Modeling of Pipeline Parallelism 2009 18th International Conference on Parallel Architectures and Compilation Techniques 2009 pp. 281-290 doi: 10.1109/PACT.2009.28.

[51] Sunil Pandey Naresh Kumar Nagwani Shrish Verma Analysis and Design of High Performance Deep Learning Algorithm: Convolutional Neural Networks International Journal of Engineering Trends and Technology 69,6 (2021):216-224.

[52] Sunil Pandey Naresh Kumar Nagwani Shrish Verma. Parallel and Scalable Deep Learning Algorithms for High Performance Computing Architectures International Journal of Engineering Trends and Technology 69(4) 236-246. (2021)

[53] Sunil Pandey Naresh Kumar Nagwani Shrish Verma. Aspects of Programming for Implementation of Convolutional Neural Networks on Multisystem HPC Architectures. Accepted in International Conference on Computational Intelligence and Energy Advancements ICCIEA-2021

[54] NEU Surface Defects Dataset http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html;

[55] K. Song and Y. Yan. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects Applied Surface Science 285 pp. 858-864 Nov. 2013

[56] Hongwen Dong Kechen Song Yu He Jing Xu Yunhui Yan Qinggang Meng. PGA-Net: Pyramid Feature Fusion and Global Context Attention Network for Automated Surface Defect Detection IEEE Transactions on Industrial Informatics 2020

[57] Yu He Kechen Song Qinggang Meng Yunhui Yan. An End-to-end Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features IEEE Transactions on Instrumentation and Measurements 202069(4)1493-1504