Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Optimization and acceleration of convolutional neural networks: A survey

Gousia Habib *, Shaima Qureshi

Department of Computer Science and Engineering, National institute of Technology Srinagar, India

ARTICLE INFO

Article history:
Received 9 July 2020
Revised 4 October 2020
Accepted 6 October 2020
Available online 20 October 2020

Keywords:
SGD
ReLu
HyPar
ADAM
NADAM

ABSTRACT

Convolutional neural networks (CNN) is a specialized case of artificial neural networks(ANN) and finds its application in computer vision and parallel distributed computing for processing of massive amount of data generated by sensors and to meet the power constraints of IOT devices. Recent advancements in parameter optimization, regularization techniques, improvement in activation functions, corresponding loss functions, advancements in the coted the research of Convolutonal Neural Network's(CNN's) in past few years. Training of neural networks is cumbersome and takes a lot of time can take days or even weeks. This limits the application of Convolutional Neural Network(CNN) in real time research fields where computational speed is of utmost importance. Thus there is a need for appropriate and enhanced computational speed to meet the requirements of these real time applications.This paper describes CNN in detail summarizes architectural evolution of CNN from 1998 to 2019. Three types of strategies have been explained to enhance the computational speed of CNN at algorithmic level and implementation level. This paper gives detailed insight about computation speed acceleration using Stochastic Gradient Decent(SGD) optimization, Fast convolution and exploiting parallelism challenges in CNN posed by these techniques and recent advancements.The paper also includes detailed view of different framework usage while implementing fast convolution or parallelism techniques. The ultimate aim of the paper to explore all such recent techniques by which we can accelerate the training speed of the CNN's without compromising the accuracy.

Contents

* Corresponding author.
E-mail addresses: gousiahabib_01phd19@nitsri.net (G. Habib), shaima@nitsri.net (S. Qureshi).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.10.004
1319-1578/© 2020 The Authors. Published by Elsevier B.V. on behalf of King Saud University.

1. Introduction

High powered and persuasive deep learning technique convolutional neural networks (CNN) identical to Artificial Neural Networks(ANNs) are composed of neurons, possessing learn able weights and biases. Input to the CNN is fed through these neurons which are responsible for performing the scalar product between inputs and learn able weights and biases. This is followed with non-linear activation function. Regular Artificial Neural Networks (ANN's) don't scale well to the full image. As the size of image is increased, it becomes impossible for the fully connected layer in the regular ANN to scale to larger images. This leads to huge increase in the number of parameters and results in over fitting of the network. Unlikely, CNN's possess 3 dimensional layer arrangement as (width, height, depth). where Depth is representing the third dimension of the activation volume and not the depth of the full neural network. The neurons in the layer are associated with the local receptive field, instead of full connectivity as in regular ANNs. The entire network maps the raw input pixels from one end to the class scores at other end through some differentiable score function thus reducing the full image into the single vector of class scores at the end of the CNN architecture. CNN also possess loss function like ANN at the last fully connected layer (e.g SVM/Softmax). But CNN forms the special case of ANN, because of its automatic learning hierarchical feature extraction process through multiple layers. Automatic feature extraction and learning features directly from raw input makes CNN applicable in a wide variety of fields like image segmentation, object detection, attention systems, speech recognition, face recognition, texture analysis, video processing and Natural language processing. The framework of CNN consists of mainly five layers, Convolutional layer (Convo layer), Pooling layer, Normalization layer and fully connected layer which are discussed in detail in the proceeding sections of the paper. Visualization of regular neural network and CNN is given below:

Fig. 1 a) Shows Regular 3 layer ANN with two hidden layers, input and output layer. A regular ANN has arrangement of neurons in single dimension giving rise to full connectivity of neurons. While, in case of convonets shown in Fig. 1b). It has three dimensional arrangement of neurons as (height, width, depth) as clearly shown by the Fig. 1b).Three dimensional input volume is received by convoNet and transformed three dimensional output volume of neuron activation. The input layer in regular Recurrent Neural Network (RNN) is represented by 3D image with its height H, width W and depth 3. Depth of 3 means image is RGB.

The huge amount of data generated every day from IoT devices, radio logical images, satellite images, and amazing improvements in hardware level, algorithmic level and structural level of CNN's have accelerated research in CNNs. Also, the utilization of different activation functions, pooling techniques, parameter optimization, and utilization of a variety of regularization methods brings huge advancement in CNN's. CNN requires less pre-processing computation as compared to hand-engineered methods such as Local Binary Patterns(LBP), Scale Invariant Feature Transform(SIFT) and Histogram of Gradient (HOG). The simple CNN architecture LeNet5 which was initially developed for object recognition using MNIST data set is given in Fig. 2.

Despite of being used two important properties that differentiates CNN from other Neural Network i.e, local receptive field and weight sharing leading to tremendously decrease in the number of parameters that greatly helps in the reduction of training time and inference.In the current era of big data, dimensionality of the input data keeps on increasing leads to complex CNN architectures for processing of such big data. This processing of ND data makes CNN computationally intensive and limits its practical implementation. The one of biggest challenges faced by the today's data science researcher's and there needs to be balance between computational speed and accuracy degradation. Which greatly enforces the current research to focus on the improvement of the computational speed and minimization of inference. The paper mainly focuses in detail on two approaches for computational speed improvement i.e, Fast convolution methods and parallelization methods. Besides this paper gives detailed evolution of CNN

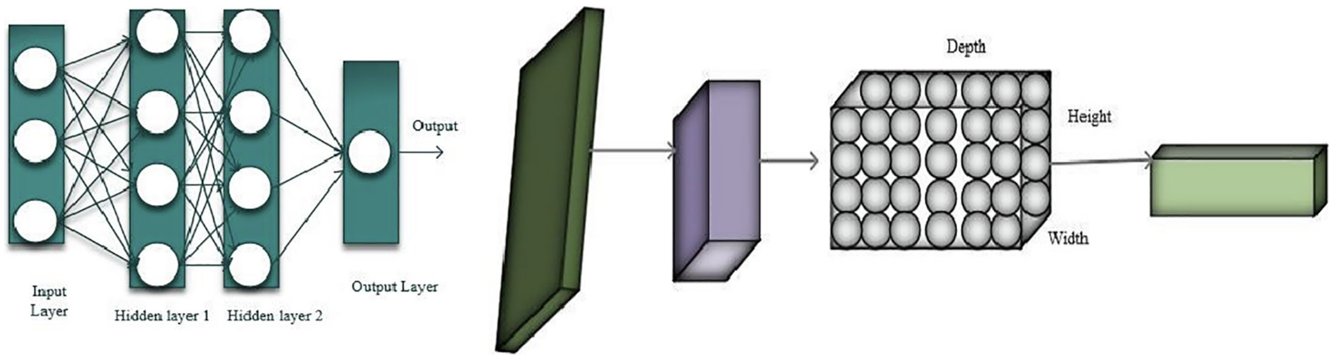


Fig. 1. a) Regular 3 layer ANN b) CovoNet with 3D input and output.

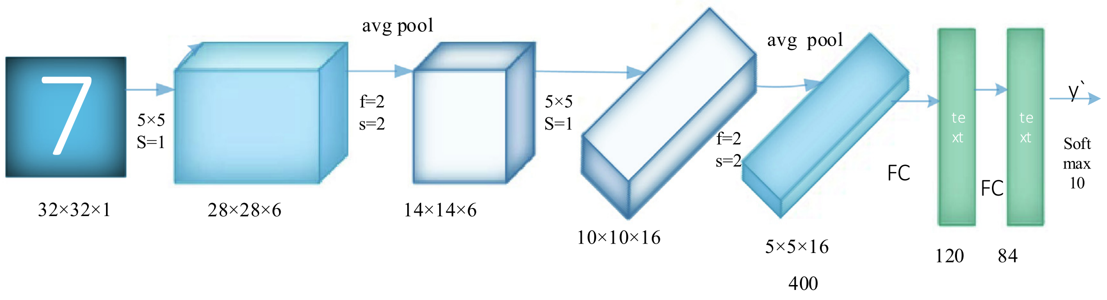


Fig. 2. Basic Lenet architecture processing an input image of size.

architectures including its components and recent advancements in CNN components and challenges posed by each of the components including convolution, pooling, activation functions etc.

2. CNN Background

The framework of CNN is based upon visual process of humans. In 1950's and 1960's (Hubel et al. 1968) explored the processing of visual responses in cat and monkey. Through various studies they found that, their visual cortex respond to smaller visual fields known as local receptive fields. Receptive fields are the region in the visual space where if stimuli occurs, it fires a single neuron. Adjacent cells use overlapping receptive fields also known as weight sharing. This is a distinctive property that makes Convolutional Neural Network(CNN) different from neural networks and helps in reducing exponential growth of parameters as the input grows (Hubel et al., 1968). In 1980, a unique concept of neocognitron was introduced by (Fukushima, k et al. 1980), that doesn't require same training weights at all the locations. Later the same was generalized and unraveled by (Le Cunn et al. 2003). They proposed 7 layer CNN for handwriting recognition sampled in 32*32 pixel images called as Le Net model shown in Fig. 3. This was the first model developed for object detection tested on MNIST dataset. But its application was limited due to huge memory requirements and processing power constraints at that time. (Pal and Pal in 1993) anticipated in their survey article that neural networks would turn oit to be broadly applied for image processing. their forecast ended up being valid. (Pal and Pal, 1993). (M.Egmont et al. 2002) reviewed an article and surveyed the application of neural systems developed for various image processing application(Egmont-Petersen et al., 2002). Until 1990s Bayesian discriminant and Parzen windows were well known. Since then neural networks have dynamically been used as another choice to great examples, classifiers and clustering techniques.

2.1. Recent encroachments in CNN

(Manrous Mohammad et al. 2015) suggested a new method for content based image classification and retrieval using optimized Pulse coupled Neural network (Mahrous Mohammed et al., 2015). Euijoon Ahn et al. 2016 devised a new algorithm based upon fusion of both deep domain transferred CNN and sparse spatial Pyramid (SSP) to extract the features from the local image dictionary. It was observed from experimental evaluation of the algorithm that this method achives the lowest error rate in comparison with existing approaches (Euijoon Ahn et al., 2016). Later on Xiang Bai et al. 2017 proposed a novel approach of CNN invariant regarded as multi-scale spatial partition network (MSP-Net) which differentiates text images by recognizing presence of text from spatial segments with varying scales of an input image from huge database of natural images. This method was able to obtain high classification accuracy and outperforms all other existing methods.Convolutional neural networks can not be limited to non-text image only but also finds its application for text based image classification technique. Besides supervised learning CNN's are capable of learning useful feature maps from huge amount of data with no labeling. General as well as specific features extracted from CNN can be transferred to universal classification task by exploring the idea of transfer learning. A lot of CNN frameworks were developed since from 1998 to 2018 like LeNet, AlexNet, Vgg 16, VGG 19, fractalNet GoogleNet, Inception module given by Google, Skip connections introduced by ResNet like Inception ResNet, ResNeXt, and some other frameworks like PolyNet, DenseNet and SeeNEt. The summary of the frameworks with top 5 error rate on MNISST, CIFAR 10 and large database ImageNet is given in the Table 5.

2.2. CNNs beyond Euclidean data classification

Besides application of CNN in euclidean data classification, CNNs are now being used in most extended field of deep learning like Generative adversarial networks (GANS) proposed by (Good fellow et al. 2014) and geometric deep learning. Geometric deep learning deals with the non-euclidean data that can't be processed by existing CNN methods. This type of data exists in several applications like social network data which can be represented by graphs in which edges denotes transfer of data and nodes denote users. Sensor networks can be modeled as graphs for distributed sensor connectivity. Gene expressions can be also models as graphs. Also in neuroscience, functional and structural anatomy of brain can be modeled using these graph models. These methods involve prep-reprocessing of data to get geometric views,then followed by CNN for further feature extraction and classification. Recently Junjie yin et al. 2020 proposed a framework 3 V-DepthPano based upon Multi-View Convolutional Neural Networks(MVCNN) (Su et al., December 2015) for training of model for extracting high precision feature maps that can be later on used for 3D shape retrieval (Yin et al., 2020). Zonghan Wu et al. 2019 put forth the detailed survey of Graph Neural Networks (GNN) with their application in particular fields. and these GNN's are broadly classified into four classes Like CNN graph neural networks, Spatial temporal graph neural networks, graph auto-encoders and RNN graph neural networks. Bronstein et al. in 2017 suggests detailed overview of application of deep learning technique in non-euclidean data which includes the description of both the graphs and man folds (Bronstein et al., 2017). Hamilton et al. in 2017 described the limited application of GNNS to combat the problem of network embedding (Hamilton et al., 2017). Dong et al. in 2019 developed 3D CNN for prediction of brain tumor using multi modal brain images and achieved the accuracy rate of 90.66 percent outperforming the existing methods (Dong et al., YYYY). Now the CNN's find its place for distributed parallel computing for processing the huge amount of data generated IOT application. Layer ise partitioning of CNN's can be used to process this massive amount of data efficiently with low power consumption and low memory for specific IoT applications. Fabiola Martins et al. in 2019 devised the deep neural network for partitioning of network for constrained IOT devices, for efficient distributed execution. It was observed that this algorithm yield maximum inference rate while minimizing the communication cost among the devices (Fabiola and Edson, 2019).

2.3. Fast convolution

Convolution operation is considered one of the most computational intensive in CNN.Many algorithms are designed to reduce the computational complexity of convolutions without degradation of accuracy.

(Yuke Wang & Keshab Parhi in 2000) Proposed popular Explicit Cook-Toom algorithm for linear convolution.Cook-Toom algorithm is one of the most important algorithm since it forms the basic building block of the large convolution algorithms.This algorithm makes use of Lagrange interpolation at $L = N + M - 1$ real number points.However,the algorithm poses huge computation cost and limits its application to only special integers.The number of operations can be significantly reduced if the L numbers are chosen carefully.This basic cook Toom algorithm was further enhanced by Yuke Wang et al. proposed generalized formula for linear convolution which interpolates at $L - 2$ non zero points.They also suggested that making use of very large scale integration(VLSI) implementation number of operations can be greatly reduced as compared to software implementations, because of 2^k representation of numbers in VLSI. which is not an option in software implementation.Basic cook-Toom algorithm is given as: if $x_0, x_1, \ldots\ldots x_{N+M-1}$ are $L = N + M - 1$ unique rel numbers.The cook-Toom algorithm calculates $D(x_i)G(x_i)$ and $S(X_i) = D(x_i)G(x_i)$ $S(X_i)$ can be interpolated from $S(x_0), S(x_1)\ldots\ldots S(x_{N+M-2})$ by using Lagrangian interpolation formula

$$S(x) = \sum_{i=0}^{L-1} S(x_i) \frac{\Pi_{j\neq i}(X - x_j)}{\Pi_{j\neq i}(x_i - x_j)} \tag{1}$$

Later on Cook-Toom algorithm was modified, makes use of popular Chinese remainder theorem for interpolation of $L - 1 = N + M - 2$ real numbers.Yuke et al. gave more generalized algorithm which requires only $L - 2 = N + M - 3$ interpolation points.The explicit formula for linear convolution given by the above mentioned authors is given in Wang and Parhi (2000) with proof as well illustrations with example.hence in their paper a general explicit formula was given which only requires $N + M - 3$ interpolation points rather than conventional algorithms for example cook-Toom algorithms requiring $N + M - 1$interpolation points.

The major drawback of the Cook-Toom algorithm and its improved version is that although number of multiplication operations were reduced but it exponentially increases the number of additions.The algorithm also becomes inefficient when the problem size get increased. For larger problems another algorithm came into existence which was given by Wino grad known as wino grads algorithm. Wino grad's algorithm is also based upon the Chinese remainder theorem(CRT) over an integer ring. The statements of the algorithm is given as:

> "It's possible to uniquely determine a non-negative integer given only its remainder with respect to the given moduli, provided that the moduli are relatively prime and the integer is known to be smaller than the product of the moduli".

(Chao Cheng & Keshab Parhi in 2004) proposed algorithm based on mixed radix algorithm and fast convolution known as iterated short convolution(ISC) algorithm. This algorithm basically decomposes the long convolutions into shorter convolution. Once the fast convolution algorithms are designed for the short convolutions then are iteratively implemented for long convolutions. The generalized algorithm for short convolutions is derived from the mixed radix algorithm which makes utilization tensor decomposition in the matrix form.Later these iterated convolutions are transformed to get fast parallel FIR filter. The efficiency of the FIR filters depends upon selectively chosen short convolution algo's. The author's observed that algorithm shows better efficiency in hardware cost reduction especially for Large FIR filters (Cheng and Parhi, 2004).

(Jaderberg Vedaldi & Zisserman in 2014) CNN brought remarkable results in the image processing and computer vision fields, but bulk processing time of convolutional layers limits their deployment in other constrained devices requiring low memory, low power consumption and high processing speed. The major cause of this bulk processing time in convolutional layers is filter redundancy. In order to eliminate this kernel redundancy in CNN. The authors mainly focus on two main ideas for substantially accelerating processing speed. This was achieved by taking advantage of cross channel or filter redundancy by constructing low rank basis of filters (actually rank 1 in spatial domain).The proposed method is independent of the underlying architecture and can be applied to all already existing Graphical Processing units(GPU's) or Central processing units(CPU's) for adaptive speed up performance.The speedup achieved by the proposed method in a real CNN which was trained for scene text character recognition is $4.5\times$ speed up with only & drop in classification accuracy. The authors also observed that if the proposed method is compared with Fast Fourier transform (FFT) the performance of the proposed method is better and achieved greater speedups (Jaderberg et al., 2014). The method can be further applicable to separable filters in layers horizontal basis layer followed vertical basis layer then finally followed by the linear combination layer.

(Cheng Tai & Tong Xiao et all in 2016) CNN's have millions of parameters which limits their applications in the IOT constrained device.If the computational cost and memory requirement of CNN's is reduced, it will be applicable to the more diverse set of applications.The mentioned authors used the concept of Low rank decomposition of tensors to eliminate the redundancy in kernels of convolution layer. Based on tensor decomposition they developed the new technique of training low rank constrained CNN's from the scratch. A significant speed up was achieved and they observed that low rank constrained CNN's showed better performance as compared to non-constrained CNN's. They analyze their performance results on best known data set CIFAR 10 and their proposed model Network in Network (NIN) achieves 93.31% accuracy without using the data augmentation. Low rank constrained CNN's achieved d times acceleration of non-constrained CNN's. The computation cost of the direct convolution is $O\left(d^2NCXY.\right)$. The computational cost of the proposed technique is $O(dK(N + C)XY)$.thus achieving acceleration of about d times. From the empirical analysis of the paper indicate that methods based upon low rank tensor decomposition can be widely used for large CNN training speed without accuracy drop (accuracy drop up to 1%) (Tai et al., 2015).

(Yulin Zhao & Donghui Wang et all in 2018) Although CNN's make noteworthy revolution in the computer vision (image and video recognition).Again limiting its application bu incurring huge computational cost.In order to speed up and reducing the computational cost they proposed hybrid algorithm based upon two existing algorithms Winograd's and Strassen algorithm they employed Visual geometry group (VGG) model to make the analysis of the results. By combination of these two algorithms they were able to save the 75%of the computational cost in comparison with the traditional algorithms. The computational performance of the proposed hybrid algorithm was compared with the original Strassen and Winograd algorithm in the Table 7 andTable 8. It was observed from the results that the proposed hybrid algorithm shows better performance for small sized matrices as far as multiplication reduction is considered but on the other side it incurs more number of additions as compared to the Strassen algorithm. The performance of the proposed algorithm is similar to Winograd's algorithm but when the matrix size is reduced, the Winograd's algorithm outperforms the hybrid algorithm (Chen et al., 2015). The major limitation of the algorithm is that although it dramatically reduces the computational cost of the CNN but it becomes difficult to implement in real time applications and embedded systems. Also makes hardware acceleration of CNN's difficult by causing hindrance in parallelizing CNN's (Zhao et al., 2018).

(Yulin Zhao & Donghui Wang et all in 2019) In order to overcome the difficulty in practical implementation of these fast convolution algorithms in real time and embedded systems mentioned above, which require lower power consumption. The authors proposed various accelerator designs using fast convolution algorithms. Proposed designs are based on field programmable gate array, requiring low power consumption.Although graphics processing unit is remarkable processor but it consumes a lot of power. In comparison with GPU Field Programmable gate array (FPGA) are widely implemented due to their flexible nature and are otherwise energy efficient.Design were implemented and evaluated using Xilinx kintex-7 32t FGPA and vivado 2014.3 using verilog. The implementation of FGPA was carried at 100 MHz. After performing their experiments they observed that all the designs almost take same amount of time to perform similar matrix convolution.As far as power analysis of different designs is considered, it was observed that dynamic power of design based on Strassen algorithm consumes power of 4% less than that of traditional algorithm. However, signal and logic powers of the design are comparatively higher as compared to traditional design.Design based on Winograd's algorithms shows 20.5% less power consumption than conventional design. Finally, the design based on Hybrid of both Strassen and Winograd's algorithm namely Strassen-Winograd algorithm outperforms all and shows total dynamic power consumption of 21.3%lesser than that of conventional designs. The authors concluded that proposed designs were able to reduce the overall resource utilization and power consumption. Further they are also able increase the performance of the designs that suits with same FPGA (Zhao et al., 2019).

(Jean Kossaifi & Adrain Bulat et all in 2019) Proposed a unified framework by linking tensor decomposition with efficient convolution. They also showed that how other computationally efficient frameworks can be derived from this unified framework. They expressed convolution as a linear combination of low rank 1 tensors and restructure the convolution efficiently. From the empirical analysis of the paper they showed that proposed algorithm can be efficiently applied to both 2D and 3D convolutions by exploiting existing frameworks. They also showed that the method is computationally efficient, memory efficient and efficient in terms of accuracy. They theoretically derived an efficient convolution algorithm for *ND* convolutions by exploiting higher order tensor decomposition (Kossaifi et al., 2019).

(Partha Maji & Andrew Mundy et all in 2019) The class of algorithms based upon popular Cook-Toom or Winograd convolution has limited application to embedded devices due to its huge computational burden. The authors tried to fulfill the gap by designing the efficient class of algorithms based upon convolution on latest Arm cortex-A CPU's that are applicable on the low constrained mobile devices. The results of their proposed method shows that shows 60% more improvement over already techniques (Maji et al., 2019).

(Di Haung & Xishan Zhang et all in 2020) As already mentioned above that one of the most popular fast convolution algorithm that is Winograd's minimal filtering algorithm used for matrix multiplication reduction to speedup CNN's. But the algorithm fails when the kernel size is larger than 1 and stride is greater than 1. For large convolution kernel sizes, it leads to the significant increase in the number of FLOPS and has accuracy drop problem. In order to combat the issues faced by the Winogard's minimal filtering algorithm. Di Haung et al. proposed modified Winograd's algorithm known as Decomposable Winograd algorithm (DWM). The method decomposes the kernels larger than 3×3 or the kernels having stride greater than into smaller kernels of size less or equal than 3×3 keeping stride as 1 and applied Winograd's algorithms to these smaller kernels. So in this way DWM helps in reduction of multiplication operation without accuracy degradation. DWM can be applicable to any size convolutional kernel which otherwise was the biggest problem in original Winograd's algorithm. DWM is able to achieve the speedup of 2without degradation in the numerical accuracy. They tested all the results on NVIDIA V 100 GPU and implemented DWM on Tensor Flow (Abadi et al. 2016) and PyTorch (Paszke et al. 2017). These platforms allows us to use DWM as a play and plug operator, makes its usage easier while inference and training. The authors observed that DWM outperforms the original Winograd's algorithm particularly in large kernel sizes. Some times DWM shows better performance as compared to CuDNN (Huang et al., 2020).

(Chao Cheng & Keshab K.Path in 2020) Proposed fast convolution algorithm for both 1D and 2D convolutions to speedup processing of CNN's by reducing matrix multiplication operation count with limited increase in number of addition operations. The CNN's efficiency can be only increased if multiplication operations are reduced at the controlled cost of increase in number of addition operations. The algorithms provide flexibility as it processes input and output feature maps with similar size and is independent of the convolution weight kernel size. The algorithm not only Speeds up computational time but also greatly improves memory access efficiency. It was observed that the proposed algorithm is able to save multiplication factor as high as 3.24 as compared to if direct 2D convolution algorithm is implemented. The potentiality of the proposed algorithm is that it is applicable to both software as well as hardware implementation. 2D parallel convolution algorithm can be simply implemented in a plug and play fashion (Cheng and Parhi, 2020).

2.4. CNN Parallelism

The outstanding achievement of CNN's in computer vision, text classification, satellite imagery and game playing have demanded increased progress in computational speedup requirement for training of CNN's. These CNN's make take weeks even days to train from the scratch on latest processors. Lot of existing parallelization techniques have came across to improve the training of CNN. Most of the techniques employ data parallelism for training acceleration of CNN's. But the technique poses some critical challenge which re already mentioned above. Another most commonly used method is Model parallelism splits the entire network into disjoint sets and each disjoint set is allocated to the dedicated device to improve the training efficiency.

(Zhihao Jia & Sina Lin et al. 2018) Proposed layer-wise parallelism allowing every layer to make utilization of individual parallelization. They allow optimization of each layer using graph search problem. They clearly define the prallelization problem using two different graphs. Device graph and computation graph, the device graphs represents all the existing hardware and how the devices are connected to each other. The latter represents how the CNN networks will be assigned device graph. Cost model was also proposed for examining the run time performance of individual parallelization strategy by exploiting dynamic graph based search algorithm to find the best parallelization strategy using proposed cost model. The layer-wise parallelism performs better than existing state of art techniques by minimizing communication cost, improving throughput and shows better scalability on addittion of numerous devices (Jia et al., 2018).

(Minji Wang & Chien-Chin Huang et al. 2018) The existing systems like TensorFLow and MXNET focuses only one of the parallelization technique at one time, requiring large sized data set to scale. The authors proposed the new approach to find the optimal tiling approach to partition the tensors with reduced communication cost among the devices. The proposed approach is basically hybrid of data and model parallelization (SOYBEAN) techniques. This SOYBEAN technique is responsible for finding automatic paralleization technique. The authors observed that SOYABEAN is $1.5\times-4\times$ better than data parallelism when implemented on AlexNet and VGG (Wang et al., 2018).

(Zhihao Jia & Matei Zaharia et al. 2019) Proposed extended search space of paralleization techniques for CNN's known as SOAP. SOAP consists of techniques that are responsible for parallelizing CNN in Sample, Operation, Attribute and parameter dimensions. They also proposed a deep learning platform FlexFlow, responsible for finding the optimal paralleization technique for a particular machine. This framework makes a prediction of best parallelization technique three times faster than previous techniques. The proposed framework was evaluated on six deep neural network models with two GPU clusters. The proposed platform was able to improve the training performance by $3.8\times$ of state of art techniques. The method was also proven to improve the scalibility on large number of devices (Jia et al., 2018).

(Yosuke Oyama & Naoya Maruyama et al. 2019) It has been observed that in model parallelism with the increase in the mini-batch size. This increase in the mini-batch size causes gradual drop in the accuracy. Therefore, mini-batch size greatly limits the maximum amount of the parallelism. This can combated by Hybrid parallelism, where maximum amount of parallelism can exceed the mini-batch size. The authors used the 3D CNN to anticipate the cosmological parameters from 3D mass distribution by exploitation of hybrid parallelism on 128 GPU's and makes utilization of 64 times greater than sample size. The proposed model allow users to make utilization of model partitioning of both sample as well as spatial dimensions for every individual layer. This brought flexibility and perfect load balancing between different GPU's. The authors successful in training acceleration of the CosmoFlow framework by exploitation of hybrid-parallelism attaining 171 T/Flop/s on 128 Tesla V 100 GPU's (Oyama et al., 2019).

(Nikoli Dryden & Naoya Maruyama al 2019) With growing size in data sets, it becomes necessary to efficiently train CNN's with reduced training time and scalable with large data set sizes. Usage of large data sets requires large amount of memory which is also one of the issue tat needs to be addressed. Default data parallelism is not the solution as it partitions samples with in mini-batch, but limits the scalability to large mini batch size and memory consumption makes it difficult for practical implementation of convolution. The authors proposed new algorithm for convolution which combines both sample as well as spatial tensor decomposition. They authors also designed performance model for CNN's and put forward method to find best parallelization strategy. The evaluated proposed algorithm on ResNet-50 for image classification tasks. They showed that their proposed method performs better and provides weak and strong scaling as compared to previous parallelization approaches and the method allows training for very large unreachable data sets (Dryden et al., 2019).

(Fabiola Martins campos de Oliveira & Edson Borin 2019) Proposed a new algorithm for partitioning neural network for efficient distributed environment execution. The proposed algorithm is responsible for optimization of inference rate and reduction of communication devices in a neural network. They used LeNet 5 for experimenting inference rate maximization in a constrained number of step. The proposed algorithm provides 38%more inferences per second for partitioning LeNet 5 than most popular partitioning offered frameworks like TensorFlow and Metis. The authors used 9 approaches for partitioning CNN for analyzing the inference rate of distributed environment into Internet of things(IOT) constrained devices. They Implemented Proposed Deep Neural Networks for constrained IOT devices DN^2PCIoT partitions neural networks presented in the form of graph in a distributed manner on multiple IOT devices aimed for achievement of maximum inference rate and communication cost minimization among various devices. The propose algorithm is also responsible for addressing the memory requirement of the network system efficiently shared by CNN parameters and biases, so that proposed framework results in valid partitioning for IOT devices. DN^2PCIoT also makes the entire system flexible and allows to make use of other objective functions as well (Fabiola and Edson, 2019).

(Chao-Tsung Huang & Yu-Chun Ding et al. 2019) As it is difficult for the CNN accelerators to support high quality image and video resolution on the edge devices due constrained bandwidth and low memory requirement and low power consumption. Therefore micro-architecture which is computationally and memory efficient for such embedded devices is of utmost importance to speed up the training inference. The authors proposed hardware based network model ERNet, for the optimization of image and video resolution based on hardware constraints. The authors also proposed coarse grained instruction set architecture, FBISA to meet the power constraints convolution by exploiting massive parallelism. The authors finally implements embedded processor of CNN (eCNN) which integrates both ERNet and FBISA with more flexible and scalable architecture. The authors showed that ERNet supports high quality and super-resolution and denoising up to 4 K Ultra-HD 30 fps while using DDR-400 and consumes 6.94 W (Huang et al., 2019).

(Li Zhou & Mohammad Hossein et al. 2019) To meet the requirement of low constrained IOT devices. The authors proposed Adaptive CNN framework that is particularly employed for optimization of heterogeneous IOT environments. The authors take the advantage of spatial partitioning together with the fusion of convolution layers, which is responsible for dynamically selecting the dynamic parallelism as per the computational resource pool available. The authors from the experiments showed that the framework outperforms the state of art techniques in terms of performance evaluation of inference speed and lowering of communication cost. The authors performed experimentation on Raspberry-Pi3 wirelessly connected devices. It was observed that proposed framework achieved the speedup of $1.9\times3.7\times$ speed up by making utilization of total of 8 devices using three different CNN models (ResNet, VGG 16, YOLO) (Zhou et al., 2019).

(Linghao Song & Jiachen Mao et al. 2020) Due to demanding use of CNN in various fields. Improvement in throughput and efficiency is of utmost importance. In order to achieve these objectives hardware acceleration is widely studied in all academic as well as industrial areas. Most of the times multiple accelerators are used to provide improved throughput and increased energy efficiency for accelerating the training of CNN's. One of the biggest challenge posed by the techniques is how to provide the best computation and data-flow between different accelerators. The authors proposed the best solution HYPAR to predict the layer wise parallelism for neural networks consisting an array of DNN accelerators. The proposed technique basically partitions the input as well as output feature maps, gradient tensor, error tensor and kernel tensors for neural network accelerators. The main motive of the optimization task is to find the best possible partition that will increase the throughput as well minimizes the communication cost between

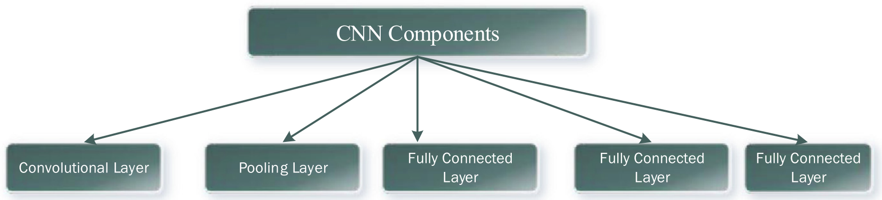


Fig. 3. Components of CNN.

the accelerators while completing the training of deep neural networks. The authors make use communication model to determine the communicate cost and hierarchical layer wise dynamic programming is used to find the partition for every individual layer. The time complexity of the proposed method is linear. Proposed model was implemented on 10 deep neural network models from basic LeNet5 to VGG 19. They reached the conclusion and stated that model parallelism is worst among all, and data parallelism is not the best parallelism technique. Hybrid-parallelism can perform better than both. Their results showed that HYPAR achieves performance gain of 3.39× and efficiency of 1.51× than default data parallelism and performance gain of 2.40× greater than "one weird trick" (Song et al., 2019).

3. WHY CNN's

We moved to CNN's from MLPs only because of two big differences, the weight sharing in CNN while performing convolution operation on input raw images. Which tremendously cut down parameter number in the whole network making network computationally less intensive. The other thing is dimensionality reduction because of introduction of poling layers in the network. Usually max pooling or average pooling is used but lot of advances has been made in these pooling techniques to accelerate the efficiency of the network. This is also responsible for induction of translational invariance and increasing the size of local receptive fields in the following CONVO layers. There is no concept of weight sharing in Multi-layer preceptrons (MLPs), therefore these networks are feasible for training of small data and large data will drastically increase the number of parameters making the network computational intensive.

4. CNN components

Simple Convolutional neural networks(CNN) can be considered as the succession of convolutional layers interleaved by pooling layers and non-linearity to make transformation of one form of activation's received at one end into another form of activation's at the other end using loss function. The CNN components are given as 3.

4.1. Convolutional layer

Convolutional layers are comprised of convolutional kernels or filters. These filters are coupled with small grid of input raw image known as local receptive field. These filters divide the entire image into small grids 3 × 3 or 5 × 5 and convolve these small grids with kernels or filters leads to the generation of feature map. Convolutional layer is responsible for extraction of locally spatial features and is responsible for doing most of the computational heavy load. The typical convolution operation is given as: $Y^k = \left(X_{i,j} * W_l^k\right)$ Where, X represents the input image (i,j) with as spatial locality W_l^k and denotes l^{th} kernel of k^{th} layer. The input to the convolutional layer is in the form $(H \times W \times C)$, where h is the height of input image, W represents width of input image and C denotes the depth of the input image. Below Fig. 4 gives the insight of typical convolutional operation. see Fig. 5–8.

Several Improvements done over CNN has brought forth CNN with better performance as compared to traditional CNN's, since from the success of deep neural network Alex Net in 2012. Some of the improvements over convolutional layer are listed in the Table 1.

4.1.1. Pooling layer

Pooling layer: Pooling layer is very important layer that is added after convolutional layer especially after ReLu layer. This layer is responsible for mitigating the problem with output feature map's sensitiveness towards location of particular features present in the input. Pooling layer makes output features more robust towards the relocation of features in the image also called translational in variance. Therefore, it induces the translational invariance in the networks and also helps in avoidance of over fitting of the

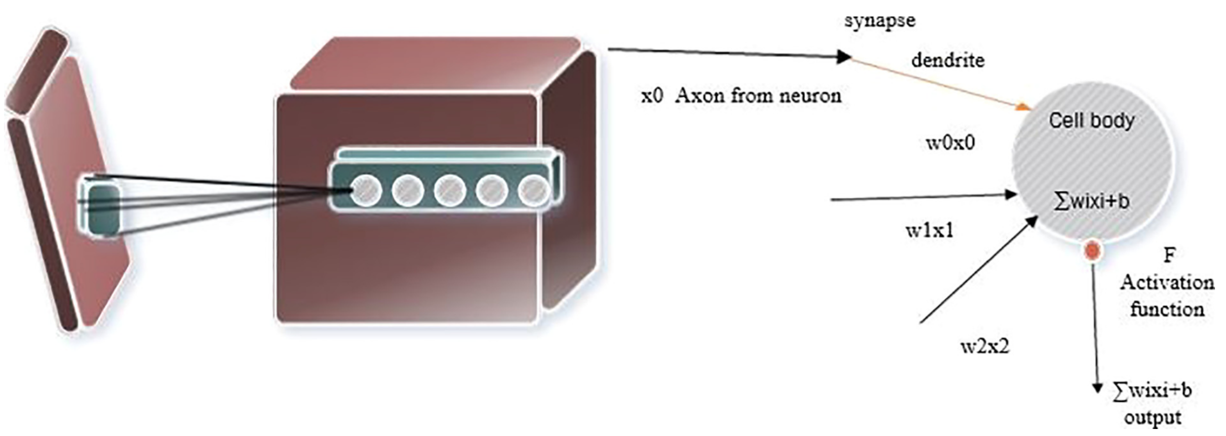


Fig. 4. Visualuation of Receptor filed in Convolutional Layer.

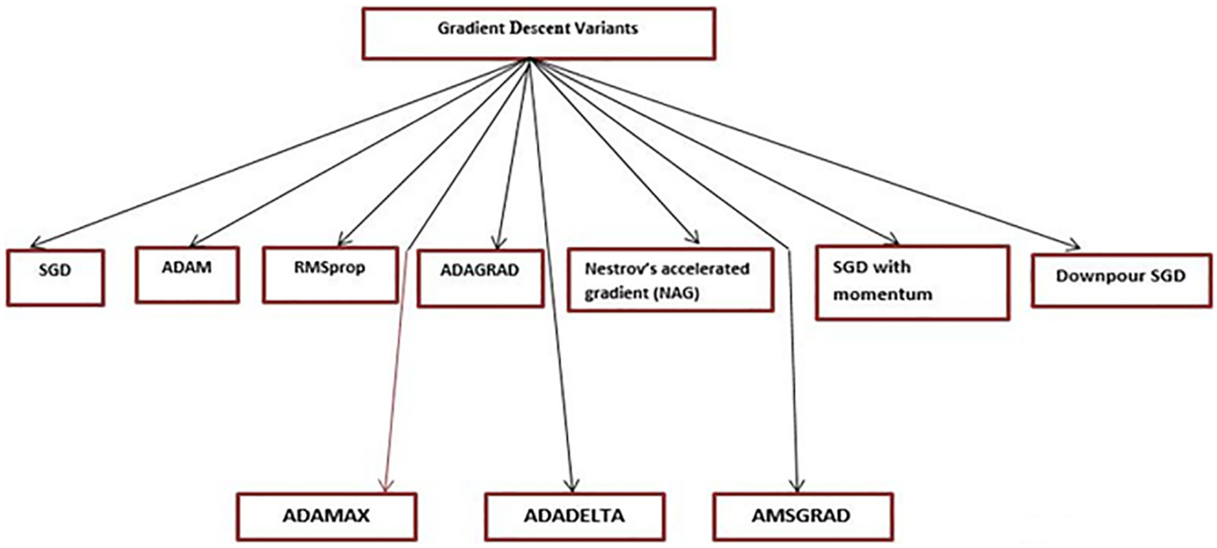


Fig. 5. Taxonomy of SGD optimizers.

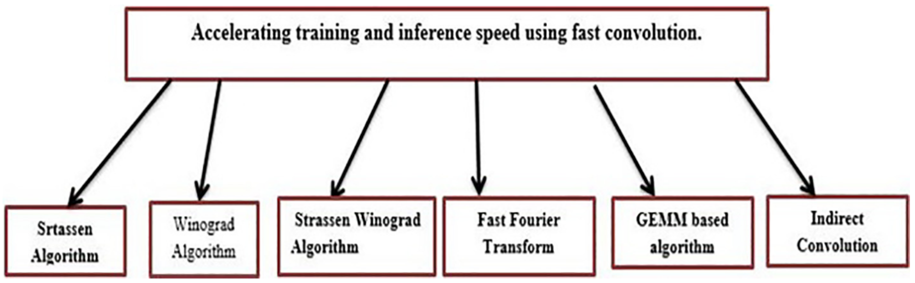


Fig. 6. Taxonomy showing training acceleration methods based on efficient convolution.

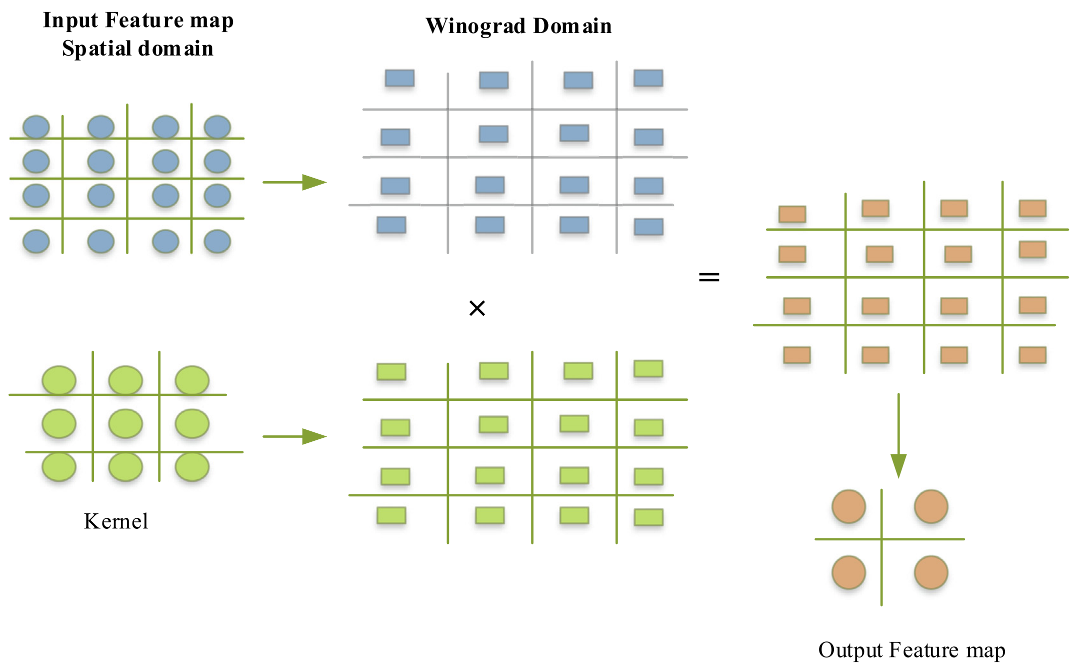


Fig. 7. Winograd Convolutions.

network. For example the input of size [224 × 224 × 64] is down sampled into output of size [112 × 112 × 64] with filter size and stride of 2. Max pooling and average pooling are two commonly used pooling techniques. These techniques face some challenges and were combated by improved pooling techniques that are summarized in table as below. Several recent pooling that is used to

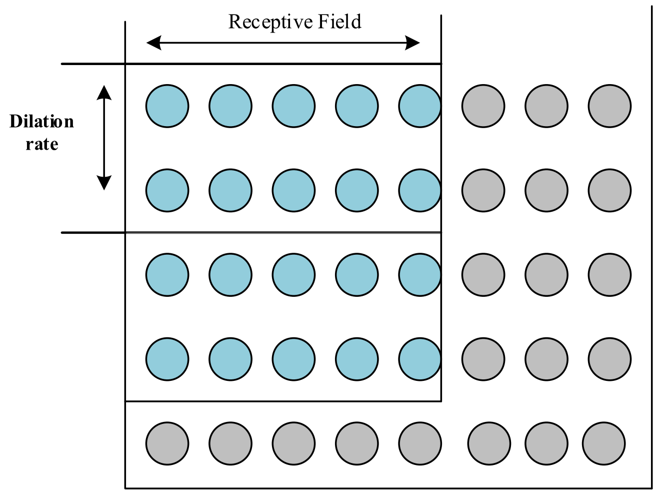


Fig. 8. Dilated Convolutions.

further improve the performance of conventional CNN's are given as in Table 2:

4.1.2. Fully-connected layer

This layer is used mostly as the last layer of the network, and possess full connectivity with all the activation's of previous layer. This layer is responsible for performing the global operation by taking input from all the various feature extraction stages and does the global analysis of the output of all the proceeding layers.

4.1.3. Activation functions

Activation functions perform the job of decision making in neural networks and helps in learning complex features from the input image. Proper choice of activation function can lead to speeding up of training process in learning complex patterns from raw input data. Various existing, as well as recent activation functions employed by CNN's, are listed in the Table 3.

4.2. Summary of CNN architectures

The summary of the CNN architectures form 1998 to 2019 are given in Table 4.

5. Training of convolution neural networks

Training of neural networks is cumbersome and takes a lot of time, sometimes training a neural network may take days or even weeks. This limits the application of deep learning system in various research fields. Because of these reasons, there is a need for efficient and improved training speed to carry out such application especially when considering the parallelization of CNN's. Stochastic gradient descent (SGD) and its variants are the most widely used algorithms for the training of deep learning models (CNN). SGD a first-order optimization algorithm finds it's applicable in a wide variety of applications but the problem with this is that it makes the exfoliation of gradient uniformly across all the directions which don't fit in the applications having poor scaling of gradient uniformity. Also in such situations, it becomes cumbersome to tune the hyper-parameter learning rate (α) during training to make sure so that it converges to the local minimum and not to move around it. Also adjusting the decay rate (α) is difficult and varies with varying datasets. Mathematically iterations of SGD are given as below:

$$W_k = W_{k-1} - \alpha_{k-1} \bigtriangledown f(W_k - 1), \tag{14}$$

where, $W_k = K^{th}$ iteration and α_k represents the learning rate. The detailed description of the SGD optimizers with challenges posed by each of the optimizer is summarized in the Table 5 and 6.

Summary

To address the problems, the solutions are given below:

1. Momentum, NAG address first issue, but NAG shows better efficiency as compared to Momentum
2. Ada grad, RMSprop addresses the second issue, but RMSprop shows better performance than Adagrad
3. ADAM, NADAM address both the issues and are better solutions.

5.1. Improving training and inference speed using fast convolution

The taxonomy of fast convolution strategies which reused to improve the training speed of convolutional neural networks is given as below:

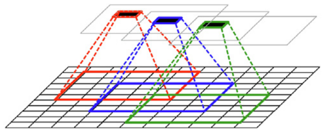
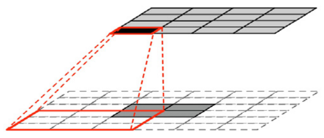
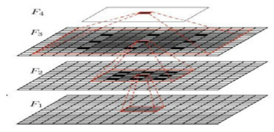
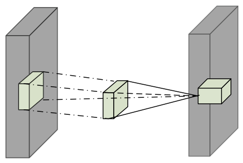
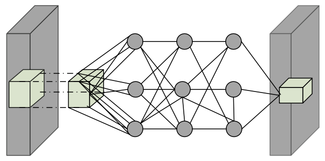
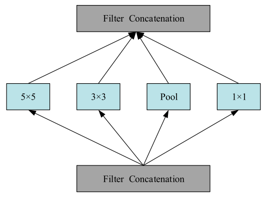
Convolutional neural networks (CNN) have been emerged as successful and widely used in computer vision, automatic driving system and other fields because of its immense success in the image processing field. CNN is complex in structure and needs a huge amount of data for processing, leads to increases in its computational burden. Therefore, limits its practical implementation for resource-constrained devices like IoT based applications and embedded system, requiring low power consumption and lesser memory requirement. Several acceleration methods have been devised to mitigate these issues and perfectly accommodate CNN into these resource-constrained applications. Denil et al. (2013) pointed out some parameter reductions in the neural network. Han et al. (2015) and Zhou et al. (2017) compressed neural networks without much degradation of accuracy. Zhang et al. (2008) utilized Low -rank approximation for efficient matrix multiplication. All the methods perform well; have their pros and cons at their place. The taxonomy of methods that are used to accelerate the training time and to improve inference speed based on efficient convolution is given in the Fig. above.

Strassen algorithm: Strassen algorithm is used to determine how matrix multiplication is performed efficiently to reduce the number of addition and multiplication operations in comparison to conventional methods. Mathematical demonstration of the Strassen algorithm is given in Yulin et al. (YYYY) and it has been shown that Strassen algorithms utilize only 7 multiplication operations instead of 8 and 18 additions in comparison with Conventional algorithms without significantly changing the computational results. The recursive versions of the Strassen algorithm can save up to $1 - \left(\frac{7}{8}\right)^N$ where N denotes the number of recursions. Architectural details of Stassen's algorithms for performing special matrix multiplication are given in Yulin et al. (YYYY). The algorithm is partitioned into three stages; its stage is responsible for transforming input data using a parameter matrix before sending it to the convolutional layer. Second stage makes initialization of 8 parallel convolutional modules. And the third stage obtains output of feature maps from these 8 parallel modules and are transformed back using parameter matrices (Yulin et al., YYYY).

Drawbacks: The method is suitable for large matrices but if the matrices are too much large then this method fails and results in degradation of the overall performance of the system. It has been observed that it performs the reduction in dynamic power consumption as compared to traditional techniques but is unable to outperform the conventional technique in terms of signal power and logic power due to its increased signal rate.

Winograd algorithm: Winograd minimal filtering algorithm that was given by Toom (Andrei, 1963) and cook (Cook, 1966) and generalization of this algorithm were done by Y. Xiong et al.

Table 1
Advancement in Convolution.

Convolution method	Description	Visualization
Tiled Convolution	Tiled CNN variant of CNN that makes CNN capable of learning rotation and scale-invariant features by tiling and multiplying different feature maps and tiled CNN's show better performance than traditional CNN's Ngiam et al., 2010	 Tiled convolution
Transposed Convolution	The reverse process of convolution operation sometimes known as deconvolution. Instead of linking different incoming activation to individual activation rather it links individual activation to many outgoing activations	 Transposed Deconvolution
Dilated Convolution	The reverse process of convolution operation sometimes known as deconvolution. Instead of linking different incoming activation's to individual activation rather it links individual activation to many outgoing activations.	 Dilated Convolution
Network in Network (NIN)	NIN given by Lin-et al makes replacement of linear filters in conventional CNN with non-linear filters or micro-networks with composite components to tackle the variance and increases the discriminative power of the local receptive field Lin et al., 2014. The initialization of micro-networks is done by multi-layer perceptron (mpconvlaye)r known as one of the best function approximation.Network in Network (NIN) composes of piling up of these micro-networks.	$a_{i},{j},{k} = \max\left(W_k^T X_{i,j} + b_k, 0\right)$  a) Linear convo layer $a_{i},{j},{k} = \max\left(W_k^T a_{i,j}^n - 1 + b_{kn}, 0\right)$  b) MpConvo Layer
Inception module	The inception module was given by Szegedy et al. Szegedy et al., 2015. It replaces the fixed filter sizes in conventional neural networks by variable filter sizes to make the network capable of capturing more specific and abstract visual patterns of variable sizes. This model was developed basically to mitigate the job of computational expense. In this module, convolutions are placed before and convolutions for further dimensionality reduction prompts an increase in the depth of CNN without making the network computational intensive.	 Filter Concatenation 5×5 3×3 Pool 1×1 Filter Concatenation Inception module

(2019) are considered as fast algorithm for performing efficient convolution. Winograd's family of algorithms is based on the Chinese remainder theorem (CRT) for polynomials. To overcome the shortcoming of the Strassen algorithm, which is applied to large convolutions? Winograd's minimal filtering algorithm outperforms for the smaller convolutions. This is based on filtering in the spectral domain and is used to reduce the computational complexity. Architectural details of Winograd algorithm is given in Winograd et al. (1980) and is divided into two stages, stage 1 initialized 8 convolutional parallel modules that are used to perform convolution operation between input feature map and kernel. Stage 2 is responsible for collecting outputs from applied convolutions. The fig. below shows 3×3 Winograd's convolutions for 2×2 output.

Drawbacks: The major challenges of this method are that the method doesn't show acceptable performance improvement when applied to dilated convolutions. To make Winograd convolutions applicable for dilated convolutions we may require additional transforms for large tiles. The figure below shows 3×3 dilated convolutions with dilation rate of 2:

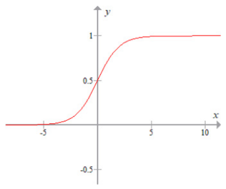
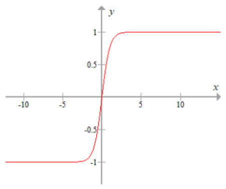
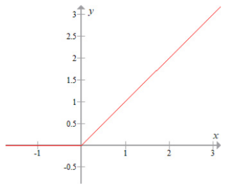
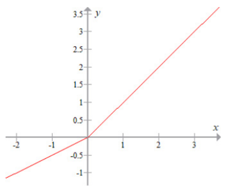
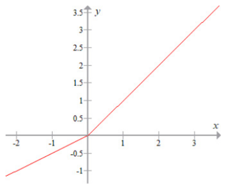
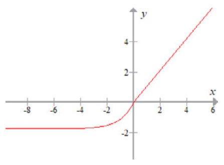
Because of the addition of zero padding's to the convolutional filter, its size gets increased. From the figure above it is clear that to obtain the 2×2 output values, 6×6 tiled transformation for dilation at the dilation rate of 2 is required. Since, Winograd transforms these zero-padded values into non zero elements in Winogrand's domain, which requires element-wise multiplication thus leading to degradation in the performance of the Winograd algorithm for dilated convolutions. Therefore, it can be concluded that the increase in the tile size can increase the number of element-wise multiplications needed for each tile leads to an increase in the computation cost for each tile. To overcome this drawback Intel (Intel, YYYY) and NVIDIA cuDNN (NVIDIA, 2014) make utilization of general matrix multiply (GEMM) for dilated convolutions. There is a lot of applications where this Winograd's minimal filtering algorithms have been used (Zhao et al., 2019). These methods

Table 2
Pooling Advancement.

Pooling method	Description	Reference
L_p Pooling	Given input feature map pooling can be defined as: $Y_{p,q,r} = \left[\sum_{(k,l)\in P_{p,q}} (b_{k,l,r})^p\right]^{\frac{1}{p}}$ (2) where $Y_{p,q,r}$ →denotes output after successful pooling operation (i,j) position in r^{th} feature vector. denotes Feature vector at in the pooling region. (1) When $(p=1)$ the above pooling refers to Average pooling which is given by the equation as: $Y_{i,j,k} = \frac{1}{\lvert P_{p,q}\rvert_{k,l\in P_{p,q}}}(b_{k,l,r})$ (3) When $P = \propto$ the above pooling refers to Max pooling which is formulated as: $Y_{p,q,r} = \max_{(k,l\in P_{p,q})}$ (4)	Pierre et al., YYYY
Mixed Pooling	Mixed pooling combines both Dropout Ngiam et al., 2010 and Drop connect Zheng et al., 2014. Mathematical formulation of mixed pooling is given as: Where, is a random number which refers to either max pooling or average pooling. Dropout: Dropout makes initialization of randomly chosen subset of activations to zero. While as, Drop connect: It instead of taking randomly chosen activations, it takes randomly chosen weights and initializes them to zero. Both of them have a great role in neural network regularization. Mixed pooling has better performance as compared to average or max pooling and helps in mitigating the problem of overfitting problems encountered by CNN's. Challenges: The major challenge of dropout lies in its restrained deployment for the convolutional layer, non-linearity layer, and pooling layer of CNN. Also, the deterministic nature of Drop connects doesn't show improved performance as compared to stochastic approaches.	Yu et al., 2014
Stochastic Pooling	Instead of taking a deterministic pooling approach (max and average) the activations are taken from a multinomial distribution made by activation in the pooling region. Using normalization of activation, Stochastic Pooling calculates for each region S_j , the probabilities P_i given as $P_i = \sum_{s\in S_j}(as)$ Sampling is done from a multinomial distribution based on P_i to choose the location k inside small region, then stochastic pooled activation is given by $X_j = ak$ where, $k \in P\left(p_1, p_2 \ldots\ldots\ldots, p_j \lvert S_j\right)$ It avoids over-fitting issues in CNN due to the presence of stochastic component as compared to max or average pooling. Challenges: Dramatic dimensionality reduction due to stochastic pooling leads to the loss of spatial information, which is combated by spectral pooling as given below.	Zeiler et al., 2013
Spectral Pooling	Fourier Spectral pooling Hubel et al., 1968 performs the down sampling of the input feature map by using low pass filtering in the frequency domain. The method poses better information retaining capability as compared to spatial pooling strategies. However, it doesn't suffer from sharp reductions in output dimensionality Challenges: The pooling process in spectral-domain is time-consuming as compared to the spatial pooling process also the introduction of imaginary processing by spectral pooling in CNN should be handled with care, otherwise, it may lead to the destruction of conjugate symmetry of Fourier transformed features of real input.	Rippel et al., 2015
Spatial Pyramid Pooling	Spatial pyramid pooling is an enhancement over a bag of words (BOW) performing pooling of structural data by using local spatial bins. Spatial bins varies directly with input size(image) and the No. of bins is independent of the size of input. Therefore several bins remain fixed. Unlike conventional deep neural networks in which the number of the sliding window depends upon input size. Therefore to increase the capability of deep neural networks to accept the image of any size, Spatial pyramid pooling layer is introduced in the neural network in place of last pooling layer. Challenges: Further improvement can be done by using multi-scale feature extraction then gathering these features using channel by channel pooling operation.	He et al., 2015
Multi-scale order less Pooling	Gong et al. Huang et al., 2018 use the idea of multi-scale orderless pooling (MOP) to enhance the geometric invariance of CNN's without adversely affecting its discerning power making CNN's robust against classification and highly variable scenes. This is achieved by performing the extraction of activation's for local and fine-grained patches using multi-scale orderless pooling with varying scales, then performing the result aggregation by using VLAD encoding Zhang et al., 2017 to generate new image representations. It has been observed that MOP-CNN is vigorous towards geometric variations as compared to conventional CNN's.	Gong et al., 2014
Ordinal Pooling	Adrein et al. [ordinal pooling] uses the concept of ordinal pooling to mitigate the adverse effects of max or average pooling strategies. Instead of taking maximum value or average value it rearranges the elements assigns different weights to different elements and then taking the weighted sum of rearranged elements in the pooling region. The type of pooling is very important in deep neural networks employing different pooling operations within the same pooling layer. It also improves accuracy over max or average pooling while speeding up the training time of CNN.	Adrien Deliège et al., 2019
Global Pooling(Global Max or Global Average pooling)	In the global pooling method entire feature map is down-sampled to a single value rather than a down-sampling input feature map. This is similar to max pooling with an assignment of pool size with input feature map size (Global max pooling). Global Average pooling in some models replace a fully connected layer to transform the input feature map into output predicted feature map of the model. In the case of global average pooling as there is no parameter for optimization, therefore we can get rid off over-fitting with the utilization of this layer. Also, it is more vigorous towards the global translations of the input.	maxpooling, YYYY]

use dedicated processing elements (PEs) for performing Winograd's convolution. But the major flaw in these methods is that for each CONV and FC there must be separate PEs, therefore, leads to critical under-utilization of resources. As this algorithm is applicable for smaller kernel CONVO layers. After the successful application of Winograd's algorithm to Alex Net, Ist, 2nd, 3rd, 4th, and 5th

Table 3
Activation Functions.

Activation Function	Mathematical formulation	Saturation	Graphical visualization	Gaps
Sigmoid	$f(x) = \frac{1}{(1+e^{-x})}$ (5)	Saturated		Vanishing Gradient Problem
Tangent Hyperbolic (Tanh)	$F(x) = \frac{2}{1+e^{-2x}} - 1$ (6)	Saturated		Vanishing Gradient Problem
Rectified Linear Unit (ReLu)	$F(x) = \begin{cases} x, & x > 0 \\ 0, & x < 0 \end{cases}$ (7)	Non Saturated		Limits its usage to only hidden layers. Sometimes leads to the dying of fragile gradients during the training process or results in dead neurons also called a dying ReLu problem.
Leaky ReLu, Parametric Rectified Linear Unit(PReLu), Randomized Leaky ReLu(RReLu)	$F(x) = \begin{cases} x & , \ x \geqslant 0 \\ \alpha x & , \ x < 0, \end{cases}$, (8) where $\alpha \in (0, 1)$	Non-Saturated		Because of linearity cannot apply for complex classification tasks. Sometimes performs worse than tan h and sigmoid functions. These may suffer from a critical over-fitting problem in CNN when applied on small datasets.
Exponential Linear Unit(ELU)	$F(x) = \begin{cases} x & , \ x \geqslant 0 \\ \alpha(e^x - 1) & , \ x < 0 \end{cases}$, (9) where $\alpha \in (0, 1)$	Non-Saturated		If x belongs to larger values then output range can drastically blow up and can go up to infinity.
Scaled Exponential Linear Unit(SELU)	$F(x) = \lambda \begin{cases} x & , \ x \geqslant 0 \\ \alpha(e^x - 1) & , \ x < 0 \end{cases}$, (10) where $\alpha \in (0, 1)$ SELU has self-normalizing properties because of zero mean and unit variance, which prompts its applications to train deeper neural networks with greater depth.	Non-Saturated		
Gaussian error Linear Unit(GELU)	$F(x) = 0.5x\left(1 + \frac{2}{\sqrt{\pi}} \int_0^{\frac{x}{\sqrt{2}}} e^{-r^2} dr\right).$ (11)			

(continued on next page)

Table 3 (*continued*)

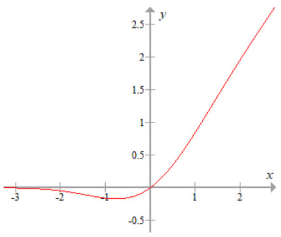
Activation Function	Mathematical formulation	Saturation	Graphical visualization	Gaps
	GELU performs better than SELU	Non-Saturated		
Max-OUT	Piece-wise linear function returning the maximum value of inputs in association with dropout regularization technique. The specializations of Max Out are ReLu and leaky ReLu. As Max Out acts as a rationalization of ReLu and leaky ReLu, therefore, eradicates the dying ReLu problem. The mathematical formulation of Max out function is given as: $$m_{i,j,k} = \max_{k\in[1,k]} W_{i,j,k} \quad (12)$$ where $W_{i,j,k}$ is the k^{th} channel of the feature vector.	Non-saturated	Challenges This function does multiply the total parameter count by two hence leads to the massive increase in the parameter count when implemented in deep neural networks.	
Prob Out	A stochastic variant of Max out is referred to as Prob Out. Instead of taking the maximum value of inputs they consider the stochastic sampling of inputs. The probabilities for every linear unit are given by: $$p_i = \frac{e^{\lambda z_i}}{\sum_{j=1}^{k} e^{\lambda z_i}} \quad (13)$$ where λ is responsible for controlling the variance of the multinomial distribution.	Non-saturated	ChallengesProb Out improves the invariance properties of max-out it is computationally expensive in the training process as compared to max Out due to huge stochastic computations.	

Table 4
CNN architecture summary.

Model	Year of release	Number of layers	No. of parameters	Error rate: Top 5	Reference
Le Net 5	1998	7	60,000	MNIST:0.9	LeCun, 2015
Alex Net	2012	8	60 M	ImageNet:16.4	Krizhevsky et al., 2012
VGG 16	2014	19	138 M	ImageNet: 7.3	Simonyan et al., 2015
Inception V_1	2014	22	5 M	ImageNet:6.7	Szegedy et al., 2014
Inception V_3	2015	48	24 M	ImageNet: 3.5	Y. Xiong et al., 2019
Inception V_4	2016	70	43 M	ImageNet: 4.01	Szegedy et al., 2016
ResNet 50	2016	50	26 M	ImageNet:5.25	https://neurohive.io/en/popular-networks/resnet/, YYYY
ResNet 152	2016	152	25.6 M	ImageNet: 3.6	He et al., 2015
Xception	2016	71	23 M	ImageNet:0.055	Chollet, 2017
FractalNet	2016	20	38.6	CIFAR-10: 7.27	Yu et al., 2016
ResNets 101	2015	101	44.6 M	ImageNet:4.60	https://neurohive.io/en/popular-networks/resnet/, YYYY
Inception ResNets	2016	572	56 M	ImageNet:3.52	https://neurohive.io/en/popular-networks/resnet/, YYYY
DenseNet	2017	190	25.6 M	CIFAR 10: 5.19	Huang et al., 2018
PolyNet	2017	–	92 M	ImageNet: 4.25	Zhang et al., 2017
Resne Xt 50	2017	50	25.5 M	ImageNet. 5.96	https://neurohive.io/en/popular-networks/resnet/, YYYY
ResNe Xt 101	2018	101	48.96 M	ImageNet: 5.59	He et al., 2016
Channel Boosted CNN	2018	–	–	–	Khan et al., 2019
See Net	2018	152	36.92 M	CIFAR: 3.58	Hu et al., 2018

Table 5
Description of SGD optimizers.

S No.	Optimization Technique	Description	Challenges
1	SGD	SGD most widely used gradient descent optimization technique for training deep neural networks. Instead of parsing all the training examples, SGD performs parameter updating on a single example and a single iteration of SGD is given as:	1). Since SGD is a first-order optimization algorithm so it does not take into consideration second-order optimization of the cost function. As the learning rate is dependent on the curvature of the function that is only taken care of by 2nd order d derivative. The steepness of the curve is measured by gradient descent while the curvature is taken of 2nd order derivative. If it is equal to 0 indicate linear curvature of the curve and in that case, the step size is equal to the learning rate. If greater than 0 then it may lead to the divergence of the curve. Due to the step size becoming lesser than the learning rate. If it is less than zero, then the step size is greater than the learning rate.
		$W_k = W_{k-1} - \alpha_{k-1} \bigtriangledown f(W_k - 1)$	2). SGD outperforms other optimization methods in terms of performance for convex optimizations while for non-convex methods to guarantee convergence of SGD is still a standing issue because of multi-local minima, saddle points, flat regions, and variable curvature.
		SGD makes the updating of parameters in the opposite direction of the negative gradient of the objective function applied for mini-batch. Step size on each iteration to reach local minima is determined by learning rate SGD makes the addition of noise to the learning process prompts improved generalization error. SGD can be applied to a wide variety of application but still, it poses some critical limitations. SGD shows better performance for convex optimization problems, but non-convex optimization to obtain a theoretical guarantee is still challenging.	3). Because the poor conditioning number of the Hessian matrix makes cost function sensitive to some directions and insensitive to other directions which makes the gradient harder to introduce big changes in the cost function.
2	SGD with momentum	The ill-conditioned Hessian matrix and when the steepness in the curvature of the curve varies in all the directions means it is no longer uniform then the performance of SGD starts degrading because of oscillations across the slope. To dampen the oscillations across slope caused by SGD an additional term to gradient descent was introduced known as momentum giving rise momentum-based SGD. The traditional momentum devised by Polyak in 1964 is regarded as the classical technique for the acceleration of gradient descent. To update the weight parameters Gradient descent with momentum not only considers the current gradient but also replaces the current gradient with the velocity vector, an exponential moving average of current and previous gradients for minimization of the objective function. Let $z(\theta)$ represents an objective function, then momentum is given as:	1). The optimization path followed by classical momentum prompts a large number of oscillations along the vertical direction of curves containing high curvature.
		$v_{t+1} = \mu_t - \varepsilon \bigtriangledown z(\theta) \quad (15)$	2). Learning rate adjustment in momentum-based SGD is challenging, too much smaller learning rate makes the gradient smaller and leads to loss of accuracy. This leads to the gradual loss of accuracy on the float-point number.
		$\theta_{t+1} = \theta_t + v_{t+1} \quad (16)$	3). Classical momentum is based on noiseless gradient estimates, and has an incompatibility with stochastic optimization leads discouragement of some researchers from being using momentum.
		Where ε represents the learning rate and is greater than zero. $\mu \in (0, 1)$ is the coefficient of momentum that controls the velocity and helps in preventing overshooting while making descent steps faster, and $\bigtriangledown z(\theta)$ represents gradient at θ_t. Polyak (1964) showed that classical momentum requires a lesser number of iterations to accelerate the convergence to the local minimum as required by the steepest descent without degradation of accuracy. The biggest advantage of SGD is that although it introduces a very smaller change to SGD but increases its speed of learning.	4). Classical CM shows intolerance to the large values of μ.
3	Nesterov's accelerated gradient(NAG)	An improvement over classical momentum having better performance is called as Nesterov's accelerated gradient. Unlike classical momentum, it calculates gradient at the point in the direction of the momentum called a look-ahead position. Nesterov's momentum first takes into consideration the previously accumulated gradient, moves in the same direction, computes the gradient and then makes the gradient updating instead of calculating the current gradient first and then moving in the updated accumulated gradient direction. This prevenient way of gradient updating helps the optimization algorithm to get rid of the overshooting of the curve problem. Like classical momentum, it is a first-order optimization algorithm with guaranteed convergence to convex functions as compared to gradient descent in the majority of situations. Mathematically NAG can be represented as:	1). Nesterov's accelerated gradient improves the global convergence rate over Stochastic gradient descent (SGD) by a factor of over smooth convex functions but behaves aggressively towards optimization of strongly non-convex functions especially with stochastic gradient signals.
		$v_{t+1} = \mu v_t - \sum \bigtriangledown f(\theta_t + \mu v_t) \quad (17)$	2). The hardest part of Nesterov's accelerate gradient is to pick up the correct learning rate and value of the coefficient of momentum to guarantee convergence. Later on, this problem was addresses y

(*continued on next page*)

Table 5 (*continued*)

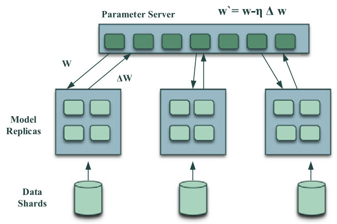
S No.	Optimization Technique	Description	Challenges
		$\theta_t = \theta_t + v_{t+1}$ (18) NAG helps in the dampening of oscillations caused during the traversing optimization path, which is one of the biggest disadvantages of classical momentum which causes large oscillations in curves with high curvature. Also, NAG is more tolerant of the large value of μ as compared to classical Momentum.	Sutsker by using manual tuning to get optimal results. Also, this problem is addressed by using an adaptive learning rate like ADAGRAD.
4	Accumulated Historical Gradient (ADAGRAD)	ADAGRAD was first proposed by Duchi et al. in 2011 that adapts the global learning rate based upon the history of the gradients. learning rate adaption is done in such a way that it makes a reduction in the learning rate of the parameters when they are receiving greater magnitude of update and increases the learning rate of the parameters when they receive the smaller magnitude of the update. in this way this guarantees convergence by accelerating parameter learning. This adaptiveness in learning by ADAGRAD helps in overcoming the problem that exists in gradient descent in which the learning rate is constant and has an impact on all the parameters of the network. The update rule of ADAGRAD is given as below: $\Delta x_t = -\frac{\eta}{\sqrt{\sum_{T=1}^{t} g_T^2}} gt,$ (19) where the denominator term in computing the L_2 norm of all the past gradients and η represents corresponding global learning rate. Which varies inversely with the magnitude of the gradients and therefore, there is a reduced learning rate for larger gradients and increased learning rate for smaller gradients. This aspect is very interesting for improving the training efficiency of various deep neural networks, in which the scale of gradients varies in different directions. One of the biggest advantages of ADAGRAD is to eradicate the manual tuning of the learning rate in the majority of deep neural networks.	1).The biggest challenge in the ADAGRAD is its sensitiveness towards the initial condition of the parameters and their respective gradients. This sensitiveness happens only when there is factoring out of magnitude in ADAGRAD. 2).Also one of the biggest loopholes in ADAGRAD is a persistent accumulation of squared gradient in the denominator term. This consistent accumulation of squared gradient makes decaying of the learning rate continuously during the entire training process until it dies up to zero which stops training of the deep neural system altogether.
5	ADADELTA	ADADELTA is an enhanced version of Adagrad, which combats the drawbacks of Adagrad, by stopping consistent decaying of learning rate during the entire training process and prevents halting of training process altogether in deep neural networks. It also eliminates the need for manual selection of tuning parameters. Ad delta uses the concept of accumulating over a window of past gradients instead of the accumulation of the sum of squared gradients during the entire duration of time. Instead of storing the squared gradients which prompt performance degradation, it makes utilization of exponentially weighted average (E) of square gradients given below as: $E\left[\nabla_\theta f\left(\theta^2\right)\right]_t = \rho E\left[\nabla_\theta f\left(\theta^2\right)\right]_{t-1} + (1-\rho)\nabla_\theta f\left(\theta_t^2\right),$ (20) where, $\left[\nabla_\theta f\left(\theta^2\right)\right]_t$ represents root exponentially weighted average sum, and ρ is decay constant. $\mathrm{RMS}\left[\nabla_\theta f\left(\theta^2\right)\right]_t = \sqrt{E\left[\nabla_\theta f\left(\theta^2\right)\right]_t + \epsilon}$ (21) The parameter update in ADADELTA is given as: $\theta_{t+1} = \theta_t - \frac{\mathrm{RMS}[\nabla_\theta]_{t-1}}{\mathrm{RMS}\left[\nabla_\theta f\left(\theta^2\right)\right]_t} \cdot \nabla_\theta f(\theta_t)$ (22)	1) The optimization path traversed by ADAdelta cause oscillations which doesn't guarantee fast convergence of the algorithm and performs worse than the momentum-based algorithm. Annealing scheduling could be added to the ADADELTA to overcome this issue. 2) The hardest task to pick up the correct learning rate. 3) The method shows sensitiveness towards the selection of learning rate. 4) The rapid diminishing of the learning rate αneeds further improvement.
6	RMS-Prop	RMS-prop was developed by Geoffrey Hinton to combat the problem of monotonically decreasing learning rate up to that extent where the training stops altogether. RMS-prop instead of all the history gradients, it only takes into consideration the small portion of gradients by decaying the rest part of history gradients. Therefore, it	1). RMS-prop lacks bias-correction terms which are important in the case of sparse gradients, if this is not corrected (value of β close to 1 causing larger step size which prompts often divergence rather than convergence.

Table 5 (*continued*)

S No.	Optimization Technique	Description	Challenges
		behaves like a moving average. RMS-prop also divides α by an exponentially decaying average of squared gradients. $E\lfloor g^2\rfloor_t = 0.9E\lfloor g^2\rfloor_{t-1} + 0.1gt^2$ (23) $\theta_{t+1} = \theta_t - \frac{\eta}{E\lfloor g^2\rfloor_t + \epsilon}gt$ (24) A typical choice of in RMS-Prop is 0.9.	
7	Adaptive moment estimation (ADAM)	ADAM can be considered as the combination of RMS-prop with momentum. So it combines both adaptive learning rates with momentum. It also records the exponentially weighted average of previous gradients similar to momentum. We can say $\text{ADAM} = \text{Momentum} \oplus \text{RMS} - \text{prop} \oplus \text{initialadjustment}.$ The update rule for Adam is given as $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$ (25) $m_t = \beta_1 m_t + (1-\beta_1)\nabla_\theta f(\theta_t)$ (26) $v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla_\theta f\left(\theta_t^2\right)$ (27) Where mean and variance It includes the bias correction mechanism by having large moments at the beginning of training. This method is particular for stochastic optimization requires little memory.	1). Despite its improvement over training time. ADAM leads to the convergence of optimal solutions in some of the applications like CIFAR −10 image classification tasks. The issue was combated b using SGD with momentum. So it has the worst classification performance as compared to SGD with momentum. 2).Wilson Pierre et al., YYYY concluded in his paper "marginal value of adaptive gradient method in machine learning that adaptive (ADAM or ADADELTA) don't improve generalization error as compared to SGD when applied to a wide variety of applications.
8	NADAM	As $\text{Adam} = \text{Momentum} \oplus \text{RMSProp}.$ Replacing momentum with Nesterov's momentum we get $\text{Nadam} = \text{Nestrov's Momentum} \oplus \text{RMSProp}.$ Before Nesterov's momentum update $\theta_{t+1} = \theta_t - (\beta m_{t-1} + \eta\nabla_\theta f(\theta_t))$ (28) $\text{Nesterov'supdaterule}\,\theta_{t+1} = \theta_t - \left(\beta m_{t-1} + \eta\nabla_\theta f\left(\theta_t - \beta_{m_{t-1}}\right)\right)$ (29) Nadam replaces m_{t-1} in momentum with m_t directly. $\theta_{t+1} = \theta_t - \left(\beta^2_{m_{t-1}} + (\beta+1)\eta\nabla_\theta f(\theta_t)\right)$ (30) As Adam $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon}(\beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta f(\theta_t))$ (31) Hence, an updated rule for NADAM is $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon}\cdot\tilde{m}_t$ (32)	
10	AMSGRAD	Mini-batches may sometimes lead to rare occurring large gradients updates which decreases their influence prompts poor convergence. To combat his issue Ams-grad is used. This makes utilization of a maximum of history squared gradients instead of taking an exponential average gradient for parameter updating.	1). There is a need for performance improvement in AMSGRAD as it has the worst performance than other methods. 2). Also, the challenging question is to detect whether convergence.

(*continued on next page*)

Table 5 (*continued*)

S No.	Optimization Technique	Description	Challenges
		$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{S}_t + \epsilon}} \cdot V_t$ (33) $\widehat{S}_t = \max\left(\widehat{S}_{t-1}, S_t\right)$ (34) $V_t = \beta_1 V_{t-1} + (1-\beta_1)\frac{\partial L}{\partial \omega_t}$ (35) $S_t = \beta_2 S_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial \omega_t}\right]^2$ (36)	
10	Downpour SGD	Downpour SGD invented by Geoffrey Hinton et al. in goggle associated with the Distbelief framework for working in a parallel distributed environment. It makes use of data parallelism and model parallelism to update the parameters. The architecture of Downpour SGD is given below as:  Parameter Server w`= w-η Δ w W ΔW Model Replicas Data Shards	However, the biggest problem with Downpour SGD is that parameters always diverge, so convergence is not fully guaranteed because in parallelization techniques there is no weight sharing between model replicas or updates with each other.

CONVO layers are successfully accelerated using this algorithm. There in summary we can say winograd's algorithm is not applicable to convolution size of greater than and stride greater than 1 (see Table 7).

Strassen-Winograd algorithm: Strassen-Winograd algorithm is the hybrid of a combination of the Strassen and Winograd algorithm to improve the computational complexity further. (Yulin Zhao et al. in 2018) combines these two algorithms to perform fast convolution operation, in which instead of recursively transforming input tiles using the kernel, they make utilization of two matrices U and V separately, which leads to the reduction of several additions introduced by this algorithm. The theoretical computational performance achieved and device utilization required by all the three algorithms for fast convolutions are given in the Table 8 as below:

From the above table, it is clear that Strassen algorithms outperform when the matrix size is small in the reduction of multiplications, but it incurs more additions as compared to the other two. Winograd shows stable performance, but Strassen shows less reduction in multiplications given small matrix size. But a lesser number of additions are incurred by this algorithm as compared to other algorithms. Once there is an increase in the matrix size Strassen-Winograd outperforms all the three algorithms including conventional one.

Recently one more solution to mitigate the problems of Winograd algorithm of not being applicable to convolution size of greater than and of stride greater than 1. If kernel size if larger than 3×3 *then In such case Winograd algorithm's precision and efficiency is compromised. (D. Haung et al. in 2020) proposed in their paper decomposable Winograd algorithm (DWM) to extend the minimal filtering algorithm to dilated convolutions where the kernel size is greater than* 3×3 *and of stride greater than 1. If the kernel size is greater than* 3×3 *DWM decomposes these larger kernels into smaller kernels then applying the Winograd's minimal filtering algorithm on these smaller kernels separately.*

Fast Fourier Transform (FFT): In convolutional neural networks two layers convo and FC layer make contributions to the network bottleneck. The earlier layer is computational intensive whereas the later layer is memory intensive. So energy efficiency and low latency acceleration of these two layers are very important. The major portion of computation is contributed by the convolutional layer. The graph in Fig. 9 shows the computation breakdown of different layers in Alex Net. It is indicated from the below graph that the Convolutional layer is most computationally intensive as compared to other layers which only contribute a small portion to the computation load of the network.The best structure of Fast Fourier transform (FFT) is depicted by the Fig. 10

In FFT methodology both the filter and data that are presented in the time domain are transformed into the frequency domain by applying fast Fourier transformation. Then FFT transformed data and filter are element-wise multiplied and finally, inverse FFT is applied to obtain the output. Therefore, convolution operation in the time domain is transformed into the multiplication operation in the spectral domain. Which is given as below:

$$W(n) * x(n) = F^{-}1[f(W(n)) \times X(n)]$$

The most popular and widely used library of FFT is the CUFFT library by NVidia and popular implementations are DSP (Nicolas Vasilache et al., 2015) and FPGA (Uzun et al., 2005). Ko et al. perform the training of CNN completely in the spectral domain with spectral-domain non-linear operations, with interpolation and Hermitian matrix symmetry. Eradicating FFT at every layer leads to training time reduction for CIFAR −10 recognition (M. Heideman et al., 1984).

Table 6
Summary of SGD optmizers.

Optimizer	Year	Learning rate α	Default value of α	Gradient	Update equation
SGD	1960	No	–	Yes	$W_k = W_{k-1} - \alpha_{k-1} f(W_k - 1)$ (37)
SGD with Momentum	1964	No	–	Yes	$\theta_{t+1} = \theta_t + v_{t+1}$ (38)
Adagrad	2011	Yes	0.01	No	$\Delta x_t = -\frac{\eta}{\sqrt{\sum_{T=1}^{t} g_T^2}} gt$ (39)
RMSprop	2012	Yes	0.001	No	$\theta_{t+1} = \theta_t - \frac{\eta}{E[g^2] + \epsilon} gt$ (40)
Adadelta	2012	Yes	0.001	No	$\theta_{t+1} = \theta_t - \frac{\mathrm{RMS}[\nabla_\theta]_{t-1}}{\mathrm{RMS}[\nabla_\theta f(\theta)]_t} \cdot \nabla_\theta f(\theta_t)$ (41)
NAG	2013	No	–	Yes	$\theta_{t+1} = \theta_t + v_{t+1}$ (42)
ADAM	2014	Yes	0.001	Yes	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$ (43)
Adam ax	2015	Yes	0.002	Yes	$\theta_{t+1} = \theta_t - \frac{\eta}{(1-\beta_1^t)(\mu_t + \epsilon)} \cdot m_t$ (44)
NADAM	2015	Yes	0.002	Yes	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} \tilde{m}_t$ (45)
AMSGrad	2018	Yes	0.001	Yes	α

Table 7
Theoretical computational performance and device utilization summary of all the three algorithms.

Matrix size	Strassen algorithm		Winograd algorithm		Strassen-Winograd algorithm	
N	MUL	ADD	MUL	ADD	MUL	ADD
2	258,048	303,104	131,072	344,176	114,688	401,520
4	1,806,336	2,416,640	1,048,576	2,294,208	802,816	2,908,608
8	1.26×10^7	1.81×10^7	8.39×10^6	1.65×10^7	5.62×10^6	2.15×10^7
16	8.85×10^7	1.31×10^7	6.71×10^7	1.25×10^8	3.93×10^7	1.62×10^8
32	6.20×10^8	9.39×10^8	5.37×10^8	9.69×10^8	2.75×10^8	1.23×10^9

Table 8
Theoretical computational performance and device utilization summary of all the three algorithms.

Algorithm	Slice Registers	Slice LUTs	Slices	DSP48E1s
Available	407,600	203,800	50,950	840
Strassen Algorithm	5543	4642	2041	252
Winograd algorithm	9544	7752	3049	128
Strassen-Winograd algorithm	8751	7396	2713	112

Table 9
FGPA based acceleration method utilizing Winograd, FFT, and GEMM transformations.

	Network Overload							
Transfor-mation	Network	Number of parameters(M)	Comp. (GOP)	Bit width	Frequency(MHZ)	Through Put (GOPS)	Memory (MB)	Power
Winograd	Alex Net C	2.3	1.3	Float 32	200	46	56.3	
	Alex Net C	2.3	1.3	Float 32	303	1382	49.7	44.3
	Alex Net C	2.3	1.3	Fixed 16	200	855	4.0	23.6
	VGG16 C	14.7	30.7	Fixed 16	200	3045	32.8	23.6
FFT	Alex Net C	2.3	1.3	Float 32	200	83	4.0	13.2
	VGG19 C	14.7	30.7	Float 32	200	123	4.0	13.2
GEMM	Alex Net C	2.3	1.3	Fixed 16	194	66	37.9	33.9
	VGG16 F	138.0	31.1	Fixed 16	200	365	14.1	25.0

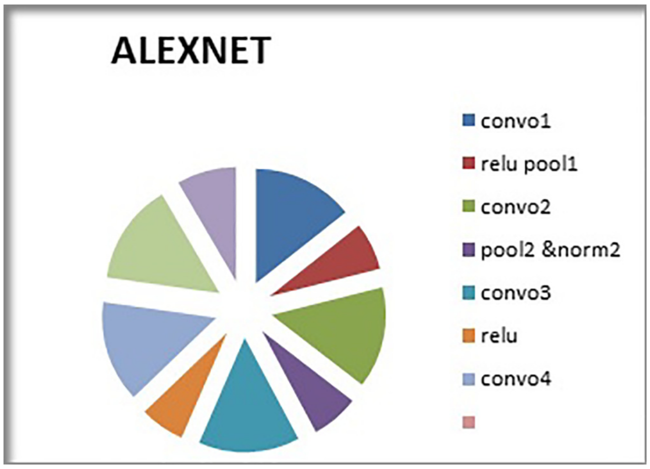


Fig. 9. Contribution of layers in computational performance.

Drawbacks: But to store FFT transformation coefficients there is a need for additional intermediate memory. This excessive memory requirement makes this approach very unpopular. And typically FFT based are used only when the size of the convolutional kernel is every close to the input image. Otherwise maybe proven less efficient for a given convolution. The computational complexity of direct convolution operation is $(n-k+1)^2k^2$, while computational complexity required by the FFT based method is $6Cn^2\log n + 4n^2$ where each FFT requires $O(n^2\log n^2) = O(2n^2\log n) = 2Cn^2\log n$ and element-wise product requires computational complexity $4n^2$. Other serious problems that must be taken care of in case of FFT based convolution are relatively low numerical precision (Zhang et al., 2018) windowing problem and extra memory overhead due to signal padding and consumes a lot of resources for performing back and forth operations while using FFT and inverse FFT (IFFT). Large receptive fields show good performance but limit convolution operations in small receptive fields.

GEMM based Algorithms: Convolution operation in GEMM based algorithms is performed GEMM (matrix -Matrix multiplications) operations. These algorithms make utilization of im2col and im2row memory transformations to transform the convolution problem into the GEMM problem. During transformation, each input pixel affecting output is copied into the matrix row which corresponds to the output. The element-wise multiplication of input patches with the kernel tensor results in the output matrix in which rows represent output pixels and columns represent output channels. Implementation of GEMM based algorithms on FPGA requires a tradeoff between computational performance, bandwidth utilization and BRAM utilization. Blocked variants of GEMM can increase the performance and make optimal use of BRAM resources and DDR bandwidth. Suda et al. (2016) and Jialiang and Jing (2017) make utilization of GEMM transformation 3D convolution to deduce Open-CL based FPGA accelerators for CNN.

Drawbacks: The problem with this method is increased memory utilization and bandwidth for the formation of the matrix patch. This overhead is directly proportional to filter size every pixel in 3×3 convolution is replicated 9 times. In overcoming these drawbacks many modifications have been done to the basic GEMM based algorithm.one of the modification is given as below:

Indirect Convolution: This algorithm is an improvement over GEMM based algorithm and performs the convolutional operations the same way as performed by the GEMM based algorithms, but possess some advantages over GEMM based algorithms which are given as below:

It helps in getting rid of costly and memory-intensive im2col operations. It has been observed by the elimination of im2col operation there is a 62% improvement over GEMM based algorithms particularly in case of convolutions possessing a smaller number of output channels.

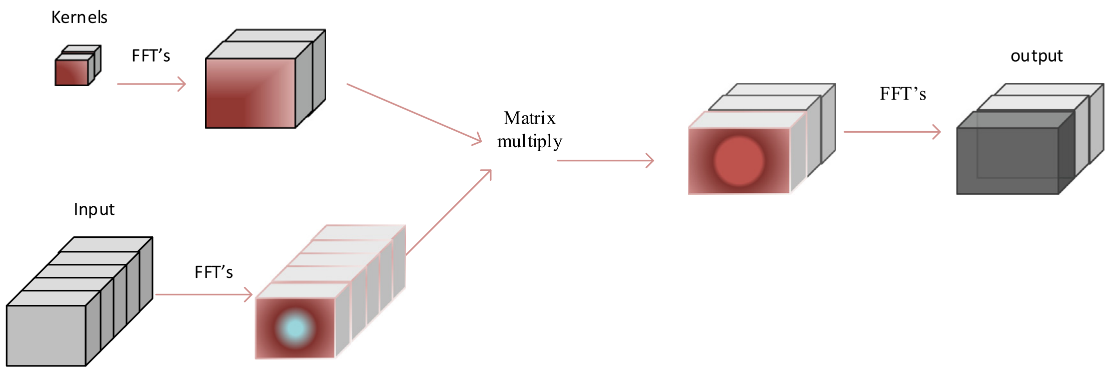


Fig. 10. FFT Algorithm.

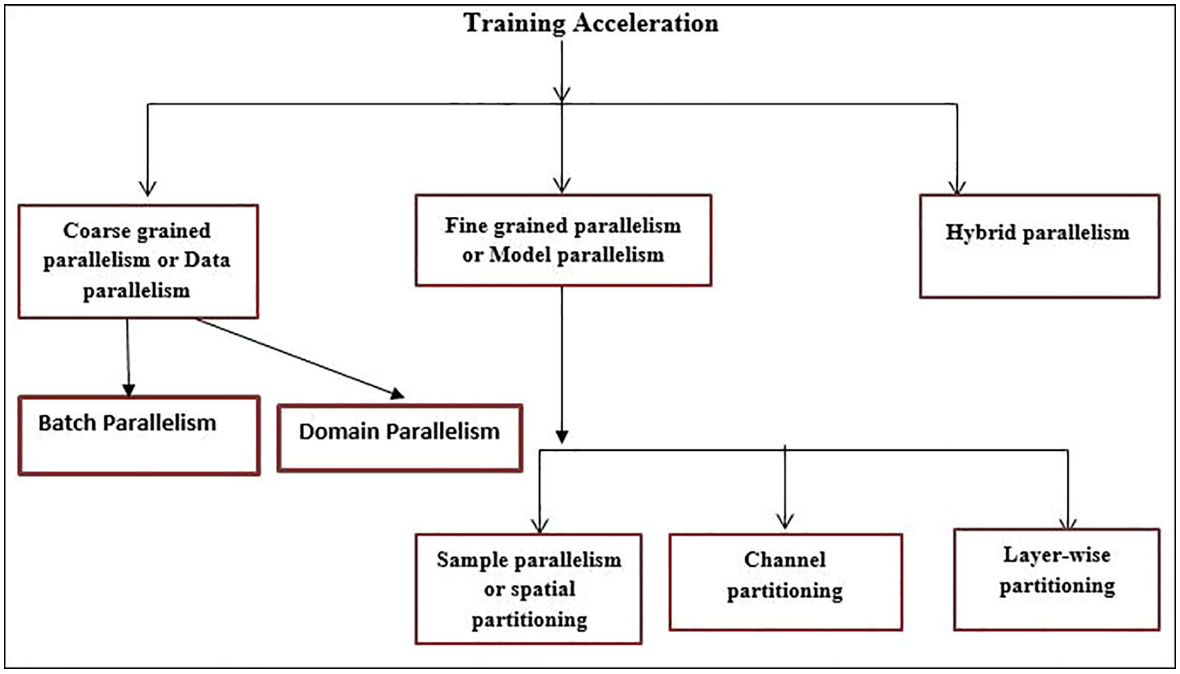

Training Acceleration
Coarse grained parallelism or Data parallelism
Fine grained parallelism or Model parallelism
Hybrid parallelism
Batch Parallelism
Domain Parallelism
Sample parallelism or spatial partitioning
Channel partitioning
Layer-wise partitioning

Fig. 11. Taxonomy of parallel strategies.

This algorithm provides the flexibility of replacing im2col buffer with indirection buffer that doesn't depend on the number of input channels, but the im2col buffer varies linearly with several input channels. Therefore, an indirect convolution algorithm is memory efficient when there is no limit on the number of input channels.

Drawbacks: Although an improvement over GEMM based algorithms poses some limitation as well which are given as:

This algorithm is particularly designed for NHWC layout with supporting libraries like tensor flow, tensor flow lite and Caffe frameworks and cannot compete with another state of art methods in terms of memory efficiency because of its striped memory access.

The algorithm works efficiently for forwarding pass of convolution but limits its application for backward pass of convolution and to transposed convolution as well.

Also, an algorithm is not efficient for depth-wise convolutions.

The FPGA based acceleration methods are summarized in Table 9, with numbers of parameters required by each methods, memory usage in MB's, bitwidth and frequency.

5.2. ***Improving training and inference speed by exploiting Parallelism***

Recent advances in convolutional neural networks like object detection (Ren et al., 2015), image classification (Han et al., 2018), Texture classification (Haralick et al., 1973), hand gesture recognition (Kim et al., 2008), biomedical image analysis (Pang et al., 2017), speech recognition (Huang et al., 2014), natural language processing (Yang et al., 2018), game playing (Stanescu et al., 2016) and other advanced CNN deep learning models requires huge amount of data for training purpose which prompts huge computational requirement to perform training of such data-hungry models. To mitigate the huge computational requirement, it is usual practice to implement distributed systems and employing clusters to parallelize the training process. Training parallelization can be achieved mainly in two ways using data parallelization and model parallelization.The taxanomy of parallelization techniques is given as in Fig. 11.

5.2.1. *Data parallelization*

(Krichevsky et al. 2012 simonyan and Zisserman 2014) Training acceleration in convolutional neural network (CNN's) is done by using data parallelism for a long. In data parallelism, the network parameters are replicated and every device (CPU or GPU) is supposed to take the same copy of the entire neural network parameters and gradient of these network parameters are calculated concerning a specific set of data samples. Then necessary aggregation of results is done using all reduce the collective operation to update the network updates at the end of one successful iteration. This kind of parallelism has been exploited in many CNN'S (Goyal et al., 2017; Akiba et al., 2018; Yamazaki et al., 2019; Milletari et al., 2016; Çiçek et al., 2016; Tran et al., 2015; Mathuriya et al., 2018; Oyama et al., 2019). Optimization of neural network parameters can be achieved using most widely. Since the granularity of this data parallelization is largely-grained therefore optimization technique known as mini-batch stochastic gradient descent optimization technique. A mathematical formulation of the above-mentioned technique is given by the equation below:

$$W^{(t+1)} = W^t - \eta^{(t)} \sum_{n-1}^{N} \bigtriangledown E_n \left(x_n \quad W^{(t)} \right), \tag{46}$$

where, $W^{(t)}$ denotes weight of neural network at time interval t. $\eta^{(t)}$ denotes learning rate, N denotes minibatch size n denotes nthsample of N. E_n Denotes Loss function, and x_n Denotes input data of nth sample

Since gradient computation $\bigtriangledown E$ can be done in parallel and independently and corresponding results are aggregated by using all reduce the collective operation to achieve parallelism of SGD. Hence in this way data-parallel training is achieved. It has been observed that collective all reduce operations require a bandwidth requirement of only O (100 Mb) which is much smaller as required by CNN computation. Data parallelism can be further divided into two types batch parallelism and domain parallelism.

Batch Parallelism In batch parallelism each process is supposed to calculate mini-batch gradient descent independently. Fist the partial sum is calculated by each individual process then all

Table 10
Data parallelism frameworks.

S. no	Framework	Toolkit Description	Accessibility
1	Tensor flow	Given by Google brain under the Apache 2.0 License in 2015 Abadi et al., 2016	Open-source
2	PyTorch	Given by Facebook Under BSD License in 2016 Tensors, 2017	Open source
3	Caffe2	Given by Berkley Research under the BSD license in 2015 Lightweight, 2016	Open source
4	MXNet	Given by Apache under license Apache 2.0 in 2015 Chen et al., 2015	Open source

reduce operaton is used to calculate mini-batch gradient descent. Communication cost in batch parallelism when particulary ring algorithm is used for performing all reduce operation is given by:

$$T_c(batch_{parallel}) = 2\sum_{i=0}^{L}\left[\alpha\lceil\log(P)\rceil + \beta\frac{P-1}{P} \mid M_i \mid\right] \tag{47}$$

where M_i denotes number of model parameters. Most of the distributed systems currently are using batch kind of parallelism.

Domain Parallelism In case of domain parallelism the input activation's are decomposed and each process is supposed to contain all the parameters of model, but instead of performing convolutions on whole images it only performs convolution operation on the subset of input image and save the subset of output activation's. If the filter size exceeds by one then halo exchange is used to communicate among boundary points. While communication some of the convolution operations can be performed that are independent of boundary data. The communication cost using domain parallelism is given by the equation below:

$$\begin{aligned}T_c(domain_{parallel}) = &\sum_{j=0}^{L}\left(\alpha + \beta B X_M^j X_c^j \lfloor f_h^j/2\rfloor\right)\\ &+\sum_{j=0}^{L}\left(\alpha + \beta B Y_M^j X_c^j \lfloor f_W^j/2\rfloor\right)\\ &+2\sum_{j=0}^{L}\left(\alpha\lceil\log(P)\rceil + \beta\frac{P-1}{P} \mid M_j \mid\right),\end{aligned} \tag{48}$$

where, $X_W^j \quad X_H^j \quad X_c^j \quad Y_W^j \quad Y_H^j \quad Y_c^j$ denotes corresponding i/p, o/p activation's width,height and channel size of j^{th} layer and $K\prime s$ denote kernel size.There is no communication overhead for convolution of size 1×1. Frameworks for achieving data parallelism are given in Table 10

This kind of parallelism is very advantageous for computational intensive deep learning models and since data parallelism possesses large granularity, therefore, requires a largely- grained inter-process mode of communication which prompts perfect load balancing. But the main problem with this kind of parallelism is that it works for deeper models with only fewer training parameters (convolutional operation parameters) and there is a gradual degradation of performance once the parameter size is increased as in case of matrix multiplication. Also, on increasing the mini-batch size beyond N value the inference accuracy of the trained neural network starts degrading which prompts to overall network performance degradation. Therefore, it limits the degree of exploitation of parallelism and turns into a scalability chokepoint for large training computing networks (distributed systems).

Table 11
Model parallelism frameworks.

S. no	Framework	Description
1	REINFORCE	Automated framework using reinforcement learning for the task- Placement to achieve model parallelism. Exploits Parallelism in single dimension i.e. operation Mirhoseini et al., 2017.
2	OptCNN	Automated framework using for deep systems possessing linear Computation graphs like VGG and ALEXNET. And also it uses parallelism in two Dimensions Sample, Attribute and operation Jia et al., 2018.
3	Tensor Flow	Open-source Toolkit gave by Google brain under the Apache 2.0 License in 2015 Abadi et al., 2016.
4	Py Torch	Open source is given by Facebook Under BSD License in 2016 Tensors, 2017.
5	Caffe	The open-source library was given by Berkley Research under the BSD license in 2015 Haralick et al., 1973.
6	DistBelief	A toolkit developed by Google. Performs acceleration of deep CNNS by Introducing model parallelism Distbelief, 2012.

Table 12
Hybrid parallelism frameworks.

S. no	Framework Description	Accessibility
1	Livermore Big Artificial neural network toolkit (LBANN) is an open-source HP-centric toolkit for training large CNN's with a huge amount of data by exploiting model parallelism with data parallelism and ensemble training techniques. Van Essen et al., 2015	Open-source
2	DIstconv is kernel library for achieving hybrid parallelism in CNN's and is supported by LBANN Dryden et al., 2019	Open source
3	Aluminum open-source GPU-Aware Library for training for large CNN's with massive dataset Dryden et al., 2018	Open source

5.2.2. Model parallelism

(Mirhoseini et al. 2017 and Kim et al. 2017) in model parallelism technique the whole neural network is partitioned into disassociated sets and each disjoint set is assigned to the dedicated processor to perform training of these disjoint network parameter set. Although it can winnow out the need for parameter contemporization amongst these dedicated devices includes the overhead of interoperation data transfer, therefore limits the parallelism amongst operations. Model parallelism can be regarded as vice versa of data parallelism in which instead of sample data filters are partitioned and corresponding feature maps. Inter-process communication takes place in forwarding propagation while there is no such communication overhead during updating of weights and backward error propagation. Different automated frameworks that are used to achieve this model parallelism are reported in the Table 11-12.

Several techniques can be used to explore this model parallelism strategy.

Spatial Partitioning: To achieve spatial partitioning or parallelism, basically spatial dimensions like height and width of input sample x are partitioned among Several dedicated devices (GPUs). As the majority of computational Operations in CNN are of nature that they don't require remote data but rely only on the neighboring data to compute the output of each layer like Outputs of convolutional and pooling layers. SO it becomes obligatory to partition a network across dimensions (spatial dimensions) to proceed with the execution of these operations concurrently on different processors without involving much communication overhead. Pooling layer parallelization can be achieved by using RELU that perform trivial parallelization irrespective of the spatial distribution The major advantage of this method is that only a few data transfer operations are required to update the parameters thus to some extent improving the overall performance of the system. Most of the operation during forwarding propagation relies only

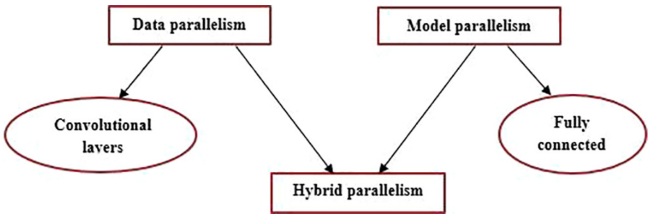


Fig. 12. Hybrid Parallelism or one weird trick (OWT).

on the local data but when the size of the filter exceeds by bottleneck (1*1) there is a need to access data that is remotely located to perform convolution operation. Thus incurring a little Communication overhead.

Channel and filter Partitioning: The convolutional and fully connected layers of CNN involve height H of input x as one dimension and W as width representing another dimension and C represents channel as third dimension giving rise to $X = H * \; W * \; C$. As these layers contain numerous channels composed of disjointed tensors whose decomposition results in channel parallelism. It becomes obligatory in current deep learning models to partition across channel dimension as the available GPU memory cant scale the number of channels presents in current models. Although the methods help us to get rid of parameter contemporization as disjoint partitioned network parameters are trained on different devices (GPUs) concurrently. But it still incurs additional communication costs because of interoperation data transfer. For achieving channel or filter parallelism data distribution of input data must be chosen carefully.

Layer-Wise Partitioning: IoT system comprises of millions of devices results in the generation of huge amount data every day. This generated data is processed by FOG may result in network overloading towards the cloud. Also, the memory constraint of CNN may limit the processing of this data on a single resource-constrained device. Therefore, it becomes necessary to partition CNN to fit resources on constrained IoT devices. Therefore, per-layer or layer partitioning of CNN‘s is employed to mitigate these resource constraint conditions. METIS, SCOTCH, DIANNE, Deep X are some of the popular frameworks that are used to achieve the layer-wise partitioning of CNN. Although, adapts CNN’s to make the execution of resource-constrained devices but it limits the execution of CNN’s on IoT devices with more critical resource constraint conditions. The above-mentioned tools if provide suboptimal results but sometimes failed to achieve the valid partitioning of the deep CNN’s.

Table 13
Comparision of Parallelization techniques.

Parameter	Data Parallelism	Model parallelism	Hybrid Parallelism
Load balancing	Factor There is perfect load balancing as far as data parallelism is concerned and can be easily maintained. It partitions sample data across multiple GPUs.	Achieving load balancing is one of the challenging tasks for Model parallelism due to the varying complexity of CNN layers.	Hybrid Task parallel applications implement CHAMELEON library for achieving load balancing. Performance improvements of 1.31X for hardware induced imbalances and 1.20X for work induced imbalanced using realistic application with AMR Çiçek et al., 2016.
Inter-GPU communication	It needs to communicate gradient information for updating of network parameters and doesn’t incur communication cost of inter - Operation data transfer. It requires more inter-GPU Cost 13.4x(model parallelism). The geometric means of inter communication cost for data parallelism is 183 GB. For the models like AlexNet, VGG-A, VGG-B, VGG-C,VGG-D and VGG-E, the communication cost ten times lesser than model parallelism Çiçek et al., 2016.	In involves transferring of Intermediate Data between Different operations so incurs inter-operation data transfer Communication cost. The geometric means of total inter communication cost in model paralleliem is 8.88 GB. This type of parallelism has higher amount of inter communication cost than data parallelism and hybrid parallelism(HYPAR).	The total inter GPU communication cost of hybrid parallelism(HYPAR) is relatively less as compared to model parallelism and extremely lesser than data parallelism. In extreme case SCONV, the totat communication cost HYPAR is equivalent to total communication cost as data parallelism and is lower as compared to model parallelism. As communication cost of data parallelism for the models AlexNet, VGG-A, VGG-B, VGG-C,VGG-D and VGG-E is ten times lower than model parallelism. Communication cost in HYPAR is another ten times lesser than data parallelism Chen et al., 2018 Communication cost for HYPAR is 0.318 GB respectively.
Training efficiency	Data parallelism effects Both training efficiency parameters like CNN convergence and CNN Accuracy. Too large mini batch size in data parallelism may cause increased utilization of GPU and at the same time degrades the overall performance of the CNN like the degradation of Accuracy. Also too small mini Batch size may underestimate the utilization of CNN. In the present era requirement of larger and deeper models is important to meet the high accuracy requirement. In these case data parallelism can not be applied. Thus it triggers the need for either model parallelism or hybrid parallelism. As the number of devices grows in case of data parallelism, it makes exponential growth of training.	Staleness issue caused in model parallelism may lead to degradation of model accuracy as well as unstable Learning. This issue is incurred only when training is performed in Pipe-lined fashion. As latter layers may process the data and compute gradients before updating by earlier layers. Larger and deeper models required for accuracy improvement can not be trained without splitting of the model thus model parallelism is very important for splitting of these deeper models. Model parallelism implements new models without restriction in the memory constraint.	Training efficiency in data parallelism can be improved by exploiting model parallelism in data parallelism. If both the techniques are combined together results in hybrid parallelism, which proves to be more efficient in minimizing the end to end training time than exploiting data parallelism alone. When HYPAR evaluated on the DNN models from LeNet to VGG’s it shows the performance gain of 3.39X and energy efficiency gain of 1.51X in comparison to data parallelism. Hybrid parallelism performs 2.40$\times$faster than trick.
Energy efficiency	Linghao et al. showed that energy efficiency of hybrid parallelism is 1.51X in comparison with data parallelism. Model parallelism is less energy efficient as compared to data parallelism. In extreme case SFC model parallelism is energy efficient as compared to data parallelism, but HYPAR has highest energy efficiency of 10.27X than model parallelism. Also they showed that HYPAR can achieve the energy efficiency between 1.03X to 1.18X in comparison with data parallelism Song et al., 2019		
Scalibility	As far as scalibity is concerned HYPAR performs better than and have highest performance gains as compared to default data parallelism after accelerators are scaled beyond 8 Song et al., 2019.		

On comparing all three partitioning techniques it can be observed that spatial partitioning achieves better inference rate as compared to channel partitioning and prompts perfect load balancing then per layer approach.

Model parallelism is beneficial over data parallelism when it comes to memory requirement as in case model parallelism memory usage varies inversely to the number of model parallelism. Therefore training can't be stopped even if only an adequate number of GPUs"s is available. This is one of the biggest advantages over data parallelism in which training abruptly stops whenever samples scale beyond N value. Model parallelism strategy is the best-suited method for that higher dimension deeper models which make use of huge GPU memory for storing higher-dimensional tensor data, for instance, 3DCNN models. Since model parallelism is a fine-grained parallelism strategy therefore, it takes into consideration only intra-layer computation and makes a presumption of the presence of data already in memory. Thus the technique fails when it comes to deciding how to achieve a parallelism strategy for multiple accelerators. So not forming the feasible solution. As it involves more data transfer operations as compared to the data-parallel technique so there is much more communication overhead, to get rid of this overhead, there is a need for careful designing of underlying architecture for the deep neural system before successful starting of training. Because of communication overhead and need for operation contemporization while computation with each layer, this kind of parallelism is not much studied.

Hybrid parallelism: To achieve a higher degree of parallelism, both data parallelism and model parallelism are combined, to constitute mixed parallelism known as hybrid parallelism and is given in Fig. 12 Due to different complexity of different layers of CNN, the Hybrid parallelism technique partitions the CNN's in such a way that convolutional layers and pooling layers' exploit data parallelism and fully connected layers' exploit model parallelism. So that deeper networks can be trained with an ample amount of resources. This is also known by the name as one weird trick in which communication cost and overhead are reduced as there is a need for gradient contemporization for fully connected layers shows a reduction in communication cost by a factor of 1.1–23.0 compared to data parallelism and model parallelism (Stanescu et al., 2016). Hybrid parallelism achieved improved runtime performance over data or model parallelism by exploiting data parallelism in Convo layers and model parallelism in FC layers. Different frameworks that are used for the exploitation of hybrid parallelism in CNN's are reported in the Table 13a.

The main problem with this kind of parallelism technique is that it makes GPU idle while the suspension of processes when processes of fully connected layers are waiting for residuals from convolutional layers. Therefore, there is a gradual degradation of efficiency of the overall system due to this idleness of GPU. It also needs remote data to proceed next computation which incurs communication costs. To mitigate this issue (Xiaoyu DU et al. in 2017) devised an alternate strategy Wheel to get rid of this idleness of GPU and to make efficient utilization of GPU (Du et al., 2017). Another fact that we have to take care of before exploiting hybrid parallelism is to carefully look after the design of the amount of parallelism across different dimensions to maintain the good scalability of the system. As compared hybrid parallelism layer-wise parallelism performs very well as it reduces communication overhead by the factor of 1.2–2.5.Overall comparison of all the three parallelization techiniques in terms of efficiency communication costetc is given by the Table 13.

6. Conclusion

After performing the detailed comprehensive literature survey and comparative analysis of CNN and its computational acceleration methods, the paper provides the detailed review about CNN, it's components and recent encroachments in both the CNN architecture as well as its components to address the computational speed issue faced by large improved recent CNN models. Current research mainly focuses on trade-off challenge between the computational speed and accuracy of large CNN models, the biggest challenges faced by the current researchers. The paper gives deep insight about the recent advanced methods to improve the computational speed issue without degradation of the accuracy. The paper includes two main approaches to address the challenges, the Fast convolution methods including Winograd's, Strassen and Winograd-Strassen method, and the other approach by exploitation of different parallelisation technique include data parallelism, model parallelism and hybrid parallelism methods. The paper summarized the different frameworks and hardware requirement utilized by each approach and what are the challenges faced by each technique's and possible solutions. Finally, the paper also gives idea about how the CNN parallelism can be implemented in a distributed fashion to meet the requirements of IOT constrained devices, embedded devices like improved computational speed and constrained memory. Besides already mentioned algorithm in the paper,some of the most recent advanced computational intelligence algorithms can be used for CNN optimization problems(minimization of computational resources and inference) like monarch butterfly optimization(MBO) (Feng et al., 2019; Feng et al., 2018), earthworm optimization algorithm(EWA) (Wang et al., 2018), elephant herding optimization(EHO) (Wang et al., 2015; Wang et al., 2016), moth search algorithm (MS) (Wang, 2018; Feng and Wang, 2018). Improved Krill herd optimization Algorithm (KH) (Wang and Yi, 2018) and artificial bee colony with information exchange optimization method.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Kudlur, M., 2016. Tensorflow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265–283).

Adrien Deliège, 2019. Maxime Istasse, Ashwani Kumar, Ordinal Pooling. In: 30th British Machine Vision conference 2019.

Akiba, T., Kerola, T., Niitani, Y., Ogawa, T., Sano, S., Suzuki, S., 2018. PFDet: 2nd place solution to open images challenge the 2018 object detection tracking. arXiv preprint arXiv:1809.00778.

Bronstein, A., Bruna, J., LCunn, Y., Szlam, A., Vandergheynst, P., 2017. Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Magazine 34 (4), 18–42.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Zhang, Z., 2015. MXnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.

Chen, C.C., Yang, C.L., Cheng, H.Y., 2018. Efficient and robust parallel DNN training through model parallelism on multi-GPU platform. arXiv preprint arXiv:1809.02839.

Cheng, C., Parhi, K.K., 2004. Hardware efficient fast parallel FIR filter structures based on iterated short convolution. IEEE Trans. Circuits Syst. I Regul. Pap. 51 (8), 1492–1500.

Cheng, C., Parhi, K.K., 2020. Fast 2D Convolution Algorithms for Convolutional Neural Networks. IEEE Trans. Circuits Syst. I Regul. Pap. 67 (5), 1678–1691.

Chollet, F., 2017. Xception: Deep learning with depth wise separable convolutions, arXiv Prepr., pp. 1610–2357.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O., 2016. 3D U-Net: learning dense volumetric segmentation from the sparse annotation. International conference on medical image computing and computer-assisted intervention. Springer, Cham, pp. 424–432.

Cook, S.A., 1966. On the minimum computation time for multiplication. Doctoral diss., Harvard U., Cambridge, Mass.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M.A., De Freitas, N., 2013. Predicting parameters in deep learning. In Advances in neural information processing systems (pp. 2148–2156).

Distbelief:framework developed by Google in 2012 https://github.com/ucla-labx/distbelief.

Dong, N., Jun Feng, L., Han, Z., Ehsan, A., Jun, W., zhengda, Y., Luyan, L., Qian, W., Jinsong, W., Dinggang, S., YYYY. Multi-Channel 3D Deep Feature Learning for Survival Time Prediction of Brain Tumor Patients Using Multi-Modal Neuroimages. In: Scientific Reports www.nature.com/scientific reports.

Dryden, N., Maruyama, N., Moon, T., Benson, T., Yoo, A., Snir, M., Van Essen, B., 2018. Aluminum: An asynchronous, GPU-aware communication library optimized for large-scale training of deep neural networks on HPC systems (No. LLNL-CONF-757866). Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States).

Dryden, N., Maruyama, N., Benson, T., Moon, T., Snir, M., Van Essen, B., 2019. Improving strong-scaling of CNN training by exploiting finer-grained parallelism. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, pp. 210–220.

Dryden, N., Maruyama, N., Benson, T., Moon, T., Snir, M., Van Essen, B., 2019. Improving stronge scaling of CNN training by exploiting finer grained parallelism. In: Proceedings of the International Parallel and distributed processing symposium IPDPS 19.

Du, X., Tang, J., Li, Z., Qin, Z., 2017. Wheel: Accelerating cnns with distributed GPUs via hybrid parallelism and alternate strategy. In: Proceedings of the 25th ACM international conference on Multimedia (pp. 393-401). ACM.

Egmont-Petersen, M., de Ridder, D., Handels, H., 2002. Image processing with neural networks–a review. Pattern Recognition 35 (10), 2279–2301.

Euijoon Ahn, R., Kumar, Ashnil, Kim, Jinman, Li, Changyang, Feng, Dagan, Fulham, Micheal, 2016. X-ray image classification using domain transferred convolutional neural networks and local sparse spatial pyramid. In: 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI).

Fabiola, M., Edson, B., 2019. Partitioning convolutional neural networks to maximize the inference rate on constrained IoT Devices.in Future Internet 2019. Ww.mdpi.com/journal/future internet.

Feng, Y.H., Wang, G.G., 2018. Binary moth search algorithm for discounted 0–1 knapsack problem. IEEE Access 6, 10708–10719.

Feng, Y., Wang, G.G., Li, W., Li, N., 2018. Multi-strategy monarch butterfly optimization algorithm for discounted 0–1 knapsack problem. Neural Comput. Appl. 30 (10), 3019–3036.

Feng, Y., Yu, X., Wang, G.G., 2019. A Novel Monarch Butterfly Optimization with Global Position Updating Operator for Large-Scale 0–1 Knapsack Problems. Mathematics 7 (11), 1056.

Gong, Y., Wang, L., Guo, R., Lazebnik, S., 2014. Multi-scale orderless pooling of deep convolutional activation features. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 392–407.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., He, K., 2017. Accurate, large minibatch SGD: Training image net in 1 hour. arXiv preprint arXiv:1706.02677.

Hamilton, W.L., Ying, R., Leskovec, J., 2017. Representation learning on graps: methods and applications. Proc of NIPs, 1024–1034.

Han, S., Mao, H., Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149.

Han, D., Liu, Q., Fan, W., 2018. A new image classification method using CNN transfer learning and web data augmentation. Expert Syst. Appl. 95, 43–56.

Haralick, R.M., Shanmugam, K., Dinstein, I.H., 1973. Textural features for image classification. IEEE Trans. Systems, Man, Cybernetics 6, 610–621.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. Pattern Anal. Mach. Intelligence (PAMI) 37 (9), 1904–1916.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep Residual Learning for Image Recognition. Multimed. Tools Appl. 77 (9), 10437–10453.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In Proc. CVPR.

https://neurohive.io/en/popular-networks/resnet/.

Huang, Z., Dong, M., Mao, Q., Zhan, Y., 2014. Speech emotion recognition using CNN. In: Proceedings of the 22nd ACM international conference on Multimedia (pp. 801-804). ACM.

Huang, G., Liu, S., Van der Maaten, L., Weinberger, K.Q., 2018. Condense net: An efficient dense net using learned group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2752–2761.

Huang, C.T., Ding, Y.C., Wang, H.C., Weng, C.W., Lin, K.P., Wang, L.W., Chen, L.D., 2019. October). eCNN: A block-based and highly-parallel CNN accelerator for edge inference. In: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 182–195.

Huang, D., Zhang, X., Zhang, R., Zhi, T., He, D., Guo, J., et al., 2020. DWM: A Decomposable Winograd Method for Convolution Acceleration. arXiv preprint arXiv:2002.00552.

Hubel et al., 1968. Science Pillani submitted to Birla institute of technology.

Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7132–7141.

Intel(R) Math Kernel Library for Deep Neural Networks (Intel(R) MKL-DNN), https://github.com/Intel/mkl-dnn.

Jaderberg, M., Vedaldi, A., Zisserman, A., 2014. Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866.

Jia, Z., Lin, S., Qi, C.R., Aiken, A., 2018. Exploring hidden dimensions in parallelizing convolutional neural networks. arXiv preprint arXiv:1802.04924.

Jia, Z., Zaharia, M., Aiken, A., 2018. Beyond data and model parallelism for deep neural networks. arXiv preprint arXiv:1807.05358.

Jia, Z., Lin, S., Qi, C.R., Aiken, A., 2018. Exploring hidden dimensions in parallelizing convolutional neural networks. arXiv preprint arXiv:1802.04924.

Jialiang Zhang, Jing Li, 2017. Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network. In: Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17, pages 25–34.

Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S., 2019. A survey of the recent architectures of deep convolutional neural networks. arXiv preprint arXiv:1901.06032.

Kim, H.J., Lee, J.S., Park, J.H., 2008. Dynamic hand gesture recognition using a CNN model with 3D receptive fields. In: 2008 international conference on neural networks and signal processing (pp. 14–19). IEEE.

Kossaifi, J., Bulat, A., Panagakis, Y., Pantic, M., Cambridge, S.A., 2019. Efficient n-dimensional convolutions via higher-order factorization. arXiv preprint arXiv:1906.06196.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. Adv. Neural Inf. Process. Syst., 1–9

LeCun, Y., 2015. LeNet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet, 20(5), 14.

A new Lightweight, modular and scalable deep learning framework https://caffe2.ai 2016.

Lin, M., Chen, Q., Yan, S., 2014. Network in network. In: Proceedings of the International Conference on Learning Representations (ICLR).

Mahrous Mohammed, Mona, Badr, Amr, Abdelhalim, M.B., 2015. Image classification and retrieval using optimized Pulse-coupled Neural Network. Expert System Appl.

Maji, P., Mundy, A., Dasika, G., Beu, J., Mattina, M., Mullins, R., 2019. Efficient winograd or cook-toom convolution kernel implementation on widely used mobile cpus. 2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2). IEEE, pp. 1–5.

Mathuriya, A., Bard, D., Mendygral, P., Meadows, L., Arnemann, J., Shao, L., Maschhoff, K., 2018. CosmoFlow: Using deep learning to learn the universe at scale. In SC18: International Conference for High-Performance Computing, Networking, Storage and Analysis (pp. 819–829). IEEE.

https://stats.stackexchange.com/questions/257321/what-is-global-max-pooling-layer-and-what-is-its-advantage-over-maxpooling-layer.

Heideman, M., Johnson, D., Burrus, C., 1984. Gauss and the history of the fast Fourier transform. ASSP Magazine, IEEE, 1(4), 14–21, Oct 1984. ISSN: 0740–7467.

Milletari, F., Navab, N., Ahmadi, S.A., 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 2016 Fourth International Conference on 3D Vision (3DV). IEEE, pp. 565–571.

Mirhoseini, A., Pham, H., Le, Q.V., Steiner, B., Larsen, R., Zhou, Y., Dean, J., 2017. Device placement optimization with reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 2430–2439. JMLR. org.

Ngiam, J., Chen, Z., Chia, D., Koh, P.W., Le, Q.V., Ng, A.Y., 2010. Tiled convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), pp. 1279–1287.

Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, Yann Lecun, 2015. Fast Convolutional Nets with fft: A GPU Performance Evaluation, in Proceedings of the International Conference on Learning Representations (ICLR), pp. 1–17.

NVIDIA cuDNN, https://developer.nvidia.com/cudnn, 2014.

Oyama, Y., Maruyama, N., Dryden, N., Harrington, P., Balewski, J., Matsuoka, S., et al., 2019. Toward Training a Large 3D Cosmological CNN with Hybrid Parallelization (No. LLNL-CONF-778764). Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States).

Oyama, Y., Maruyama, N., Dryden, N., Harrington, P., Balewski, J., Matsuoka, S., Snir, M., Nugent, P., Van Essen, B., 2019. Towards training a large 3D cosmological CNN with hybrid parallelization (No. LLNL-CONF-778764). Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States).

Pal, N.R., Pal, S.K., 1993. A review on image segmentation techniques. Pattern Recognition 26 (9), 1277–1294.

Pang, S., Yu, Z., Orgun, M.A., 2017. A novel end-to-end classifier using domain transferred deep convolutional neural networks for biomedical images. Computer Methods Programs Biomedicine 140, 283–293.

Pierre, S., Soumith, C., Yann, L., YYYY. Convolutional Neural Networks Applied to House Numbers Digit Classification, in: http://arxiv.org/abs/1204.3968v1.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-CNN: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems (pp. 91–99).

Rippel, O., Snoek, J., Adams, R.P., 2015. Spectral representations for convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), pp. 2449–2457.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. ICLR 75 (6), 398–406.

Song, L., Mao, J., Zhuo, Y., Qian, X., Li, H., Chen, Y., 2019. Hypar: Towards hybrid parallelism for deep learning accelerator array. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 56–68.

Stanescu, M., Barriga, N.A., Hess, A., Buro, M., 2016. Evaluating real-time strategy game states using convolutional neural networks. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, pp. 1–7.

Naveen Suda, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, Yu Cao. 2016. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks. In: Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays- FPGA '16, pages 16–25.

Su, H., Maji, S., kalogerakis, E., et al., December 2015. Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE conference on computer vision, Santiago, Chile, pp. 945–953.

Szegedy, C. et al., 2014. Going Deeper with Convolutions, arXiv:1409.4842.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9.

Szegedy, C., Ioffe, S., Vanhoucke, V., 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv Prepr. arXiv1602.07261v2, vol. 131, no. 2, pp. 262–263.

Tai, C., Xiao, T., Zhang, Y., Wang, X., 2015. Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067.

Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://pytorch.org 2017.

Toom, Andrei L, 1963. The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady 3, 714–716.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., 2015. Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp. 4489–4497.

Uzun, I.S., Amira, A., Bouridane, A., 2005. FPGA implementations of fast Fourier transforms for real-time signal and image processing. In Vision, Image and Signal Processing, IEE Proceedings-, volume 152, pages 283–296. IET.

Van Essen, B., Kim, H., Pearce, R., Boakye, K., Chen, B., 2015. LBANN: Livermore big artificial neural network HPC toolkit. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments (p. 5). ACM.

Wang, G.G., 2018. Moth search algorithm: a bio-inspired meta-heuristic algorithm for global optimization problems. Memetic Comput. 10 (2), 151–164.

Wang, Y., Parhi, K., 2000. Explicit Cook-Toom algorithm for linear convolution. In 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100) (Vol. 6, pp. 3279–3282). IEEE.

Wang, H., Yi, J.H., 2018. An improved optimization method based on krill herd and artificial bee colony with information exchange. Memetic Comput. 10 (2), 177–198.

Wang, G.G., Deb, S., Coelho, L.D.S., 2015. Elephant herding optimization. In: 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI). IEEE, pp. 1–5.

Wang, G.G., Deb, S., Gao, X.Z., Coelho, L.D.S., 2016. A new meta-heuristic optimization algorithm motivated by elephant herding behavior. Int. J. Bio-Inspired Computation 8 (6), 394–409.

Wang, G.G., Deb, S., Coelho, L.D.S., 2018. Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems. Int. J. Bio-Inspired Computation 12 (1), 1–22.

Wang, M., Huang, C.C., Li, J., 2018. Unifying data, model and hybrid parallelism in deep learning via tensor tiling. arXiv preprint arXiv:1805.04170.

Winograd, S., 1980. Arithmetic Complexity of Computations. SIAM, Philadelphia, PA, USA.

Yamazaki, M., Kasagi, A., Tabuchi, A., Honda, T., Miwa, M., Fukumoto, N., Nakashima, K., 2019. Yet Another Accelerated SGD: ResNet-50 Training on ImageNet in 74.7 seconds. arXiv preprint arXiv:1903.12650.

Yang, L., Dong, P.Z., Sun, B., 2018. U.S. Patent No. 10,083,171. Washington, DC: U.S. Patent and Trademark Office.

Yin, J., Ningning, H., Jing, T., Meie, F., 2020. Recognition of 3D shapes based on 3V-depthpano CNN. In: Mathematical problems in engineering volume 2020, article id 7584576.

Dingjun Yu, Hanli Wang, Peiqiu Chen, Zhihua Wei, 2014. Mixed Pooling for Convolutional Neural Networks, in: Springer International Publishing Switzerland 2014. RSKT 2014, LNAI 8818, pp. 364–375, 2014.

Yu, L., Yang, X., Qin, J., Heng, P.A., 2016. 3D FractalNet: dense volumetric segmentation for cardiovascular MRI volumes. In: Reconstruction, segmentation, and analysis of medical images. Springer, Cham, pp. 103–110.

Yulin, Z., Donghui, W., Leiou, W., Peng, L., YYYY. A faster algorithm for reducing the computational complexity of convolutional neural network.in: algorithms MPDI.

Xiong, Y., Kim, H.J., Hedau, V., 2019. ANTNets: Mobile Convolutional Neural Networks for Resource Efficient Image Classification.

Zeiler, M.D., Fergus, R., 2013. Stochastic pooling for regularization of deep convolutional neural networks. In: Proceedings of the International Conference on Learning Representations (ICLR).

Zhang, K., Tsang, I.W., Kwok, J.T., 2008. July). Improved Nystrom low-rank approximation and error analysis. In: Proceedings of the 25th international conference on Machine learning, pp. 1232–1239.

Zhang, X., Li, Z., Change Loy, C., Lin, D., 2017. Polynet: A pursuit of structural diversity in very deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 718–726.

Jiyuan Zhang, Franz Franchetti, Tze Meng Low. 2018. High performance zero-memory overhead direct convolutions. arXiv preprint arXiv:1809.10170.

Zhao, Y., Wang, D., Wang, L., Liu, P., 2018. A faster algorithm for reducing the computational complexity of convolutional neural networks. Algorithms 11 (10), 159.

Zhao, Y., Wang, D., Wang, L., 2019. Convolution accelerator designs using fast algorithms. Algorithms 12 (5), 112.

Zhao, Y., Wang, D., Wang, L., 2019. Convolution Accelerator Designs Using Fast Algorithms. Algorithms 12 (5), 112.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L., 2014. Time series classification using multi-channels deep convolutional neural networks. In: Proceedings of the International Conference on Web-Age Information Management (WAIM), pp. 298–310.

Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y., 2017. Incremental network quantization: Towards lossless CNN's with low-precision weights. arXiv preprint arXiv:1702.03044.

Zhou, L., Samavatian, M.H., Bacha, A., Majumdar, S., Teodorescu, R., 2019. Adaptive parallel execution of deep neural networks on heterogeneous edge devices. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, pp. 195–208.