**Optimized Implementation of Convolution Neural Network**

**Abstract**

The Convolution Neural Networks have several different filters (also known as kernels) that consist of parameters that can be trained. Convolution Neural Network convolves any given image by the user spatially. It detects characteristics such as shapes, edges and corners.The large number of filters are highly efficient and effective. They learn to extract spatial features from any given image that are certainly based on the learned weights by back propagation approach. Layers of filters when stacked can be used to detect the spatial shapes which have a high level of complexity from the spatial features at every subsequent level. Therefore, the layers of filters can successfully extract the characteristics of a given image by considering the edges and vertices into an abstracted representation of high quality. Patterns in pixel values are read and extracted from the given input images in Dense Networks.

In most of the deep learning techniques, algorithms and data training is time consuming. In training neural networks most of the computation is spent on floating point multiplications. In our project, we apply an approach to training that eliminates the need for most of these. Our method consists of two parts:

First is Sparse Ternary Connect and Second one is Dual indexing module using these two parts

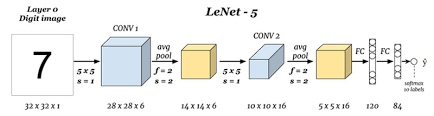
we make a simple network which reduces computations. Experimental results across a popular datasets MNIST show that this approach

not only does not hurt classiﬁcation performance but can result in even better performance than

standard stochastic gradient descent training, paving the way to fast, hardware friendly training of neural networks.

**Introduction**

A Convolutional neural network (CNN) is a deep learning approach which can take an image as an input. It then assigns important learnable or trainable weights and biases to various objects in the image and is able to classify problems and other applications. Convolutional neural network is composed of multiple building blocks, such as convolutional layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm. The preprocessing required in a Convolutional neural network is very less as compared to other algorithms. Convolutional neural networks have the ability to learn filters(Weight and biases). A Convolutional neural network is able to successfully capture the important feature in an image through the application of relevant filters. The architecture performs a better fitting to the datasets of images due to the reduction in the number of parameters involved and reusability of weights.



In recent years, deep learning has been used extensively in a wide range of fields. In deep learning, Convolutional Neural Networks give the most accurate results in solving real world problems such as computer vision, mainly in face recognition, scene labelling, image classification, action recognition, human pose estimation and document analysis. CNN is also used in speech recognition and text classification for natural language processing.

Training deep neural networks has long been computational demanding and time consuming and requires more storage capacity and any spatial hardware. For some state-of-the art architectures, it sometimes takes weeks to get train models. Another problem is that the demand for memory can be large. For example, many models in speech recognition or machine translation need 10 Gigabytes or more of storage requirement. To deal with these issues it is common to train deep neural networks by an optimized method which we proposed or by resorting to GPU or CPU clusters and to well designed parallelization strategies .

In training a network most of the computation performed are floating point multiplications. In our project, we have tried to put more focus on eliminating most of these multiplications to reduce computation. In our project, we have combined two existing methods for optimizing CNN. These are :

1. Sparse Ternary Connect(STC) -

The proposed algorithm trains the network so that the weights are represented using a structured sparse ternary format. This format allows +1 or -1 only at specified locations, while most of the values are pruned to zero.

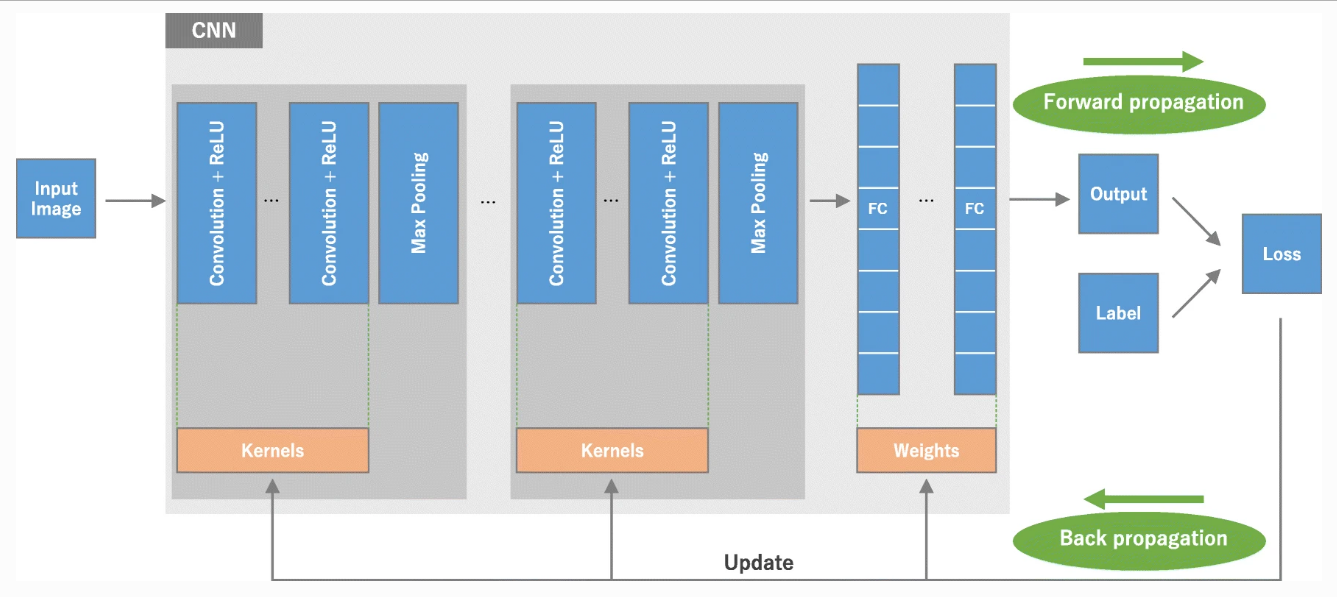
2.Dual Indexing Module(DIM) -

In convolutional neural networks, sparsity is widely observed by zeroing a large portion of both activations and weights without impairing the result, by keeping the data in a compressed-sparse format. Adding an indexing mechanism could facilitate the handling of the compressed-sparse format. Cnvlutin removes the zeros in activations and stores the memory offsets of non-zero data for rapid selection of matched weights. To reduce these calculations.

**Background and Literature Review**

**Background**

A CNN is a deep learning model used for processing data that has a grid pattern, such as images. Convolutional Neural Network is composed of multiple building blocks, they are Convolution layers, Pooling layers and Fully connected layers. The first two, i.e., convolution layer and pooling layers are used to perform feature extraction, whereas the third, i.e., a fully connected layer extracts features into final output, known as classification. A convolution layer plays a main role in CNN, which is composed of a stack of mathematical operations, such as convolution, which performs a special type of linear operation. In digital images, values of pixels are stored in the form of a two-dimensional grid, i.e., an array of numbers , and a small grid of parameters known as kernel, a feature extractor, is applied at each position of image, which makes CNNs highly efficient for image processing. As one layer provides its output to the next layer, so extracted features become more complex. The process of optimizing parameters such as kernels is called training, which is used to perform minimized difference in outputs and truth labels from an optimization algorithm known as backpropagation and gradient descent.



Above figure shows the architecture of a convolutional neural network (CNN) and the process of training. A model performance for kernels and weights is calculated from forward propagation on training dataset. Kernels and weights are updated with the help of loss value through backpropagation with the help of gradient descent algorithm using ReLU (rectified linear unit).

**Literature review**

Following articles provide the basic concepts necessary to design a CNN architecture, as well as providing the state-of-the-art energy efficient CNN designs and commenting on the future trends that still need to be addressed.

• **Neural networks with few multiplications:[1]**

For most deep learning algorithms training is notoriously time consuming. Since most of the computation in training neural networks is typically spent on floating point multiplications, we investigate an approach to training that eliminates the need for most of these. Our method consists of two parts: First we stochastically binarize weights to convert multiplications involved in computing hidden states to sign changes. Second, while back-propagating error derivatives, in addition to binarizing the weights, we quantize the representations at each layer to convert the remaining multiplications into binary shifts. Experimental results across 3 popular datasets (MNIST, CIFAR10, SVHN) show that this approach not only does not hurt classification performance but can result in even better performance than standard stochastic gradient descent training, paving the way to fast, hardware-friendly training of neural networks.

• **A Novel Zero Weight/Activation-Aware Hardware Architecture of Convolutional Neural Network:[2]**

It is imperative to accelerate convolutional neural networks (CNNs) due to their ever-widening application areas from server, mobile to IoT devices. Based on the fact that CNNs can be characterized by a significant amount of zero values in both kernel weights and activations, we propose a novel hardware accelerator for CNNs exploiting zero weights and activations.

• **Design Space Exploration of FPGA-Based Deep Convolutional Neural Networks:[3]**

Deep Convolutional Neural Network (CNN) has been recognized as the most effective model for pattern recognition and classiﬁcation tasks. With the fast growing Internet of Things (IoTs) and wearable devices, it becomes attractive to implement DCNNs in embedded and portable systems. However, novel computing paradigms are urgently required to deploy DCNNs that have huge power consumptions and complex topologies in systems with limited area and power supply. Recent works have demonstrated thatStochastic Computing (SC) can radically simplify the hardware implementation of arithmetic units and has the potential to bring the success of DCNNs to embedded systems.

• **Supporting Compressed-Sparse Activations and Weights on SIMD-like Accelerator for Sparse Convolutional Neural Networks:[4]**

Sparsity is widely observed in convolutional neural networks by zeroing a large portion of both activations and weights without impairing the result. By keeping the data in a compressed-sparse format, the energy consumption could be considerably cut down due to less memory traffic. However, the wide SIMD-like MAC engine adopted in many CNN accelerators cannot support the compressed input due to the data misalignment.

• **Sparse Ternary Connect: Convolutional Neural Networks Using Ternarized Weights with enhanced Sparsity:[5]**

Convolutional Neural Networks (CNNs) are indispensable in a wide range of tasks to achieve state-of-the-art results. In this work, we exploit ternary weights in both inference and training of CNNs and further propose Sparse Ternary Connect (STC) where kernel weights in float value are converted to 1, -1 and 0 based on a new conversion rule with the controlled ratio of 0.

**Importance of the Project**

Convolution neural networks (CNN) are known for its high accuracy for image recognition. In recent years, it has been widely used in many image-related machine learning algorithms. CNN consists of a large number of computations and it is essential to accelerate the CNN computation by a hardware accelerator, which can be FPGA, GPU and ASIC designs. However, CNN accelerator has a drawback: the large time and power consumption caused by the data access of off-chip memory.

Optimization of CNN will help in various fields of technologies. It can be a great support to the Cyber Security and Automobile industry.

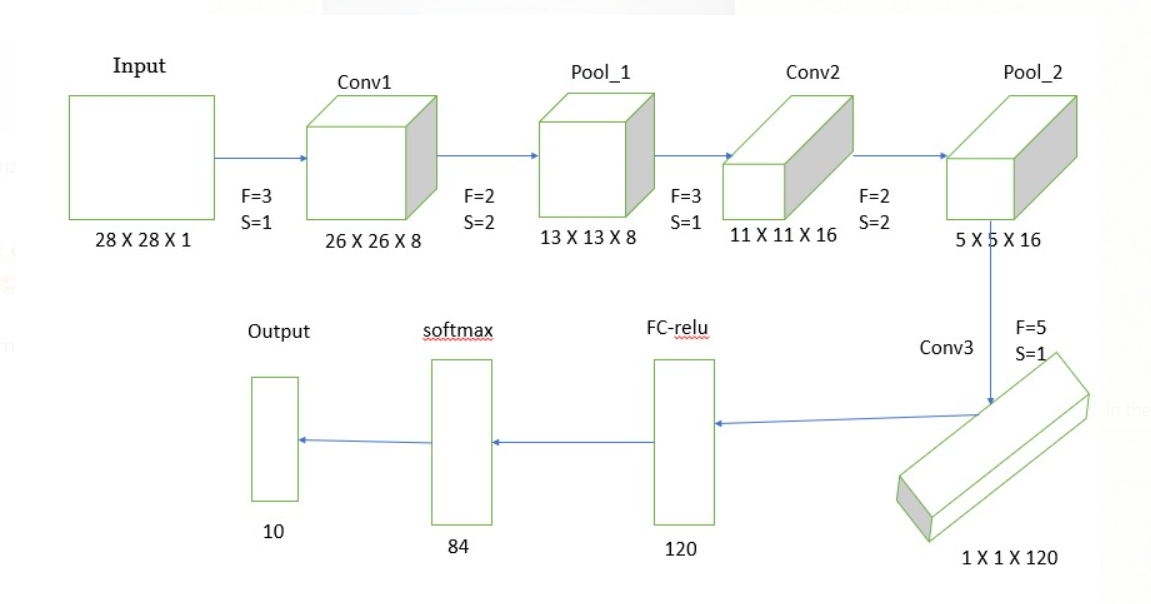
CNN has the following advantages:

* Weight sharing
* Memory Saving
* Independent of local variations in Image
* Equivariance

**Proposed Method**

The architecture we used in this work is LeNet. The LeNet architecture is an excellent “first architecture” for Convolutional Neural Network (especially when trained on the MNIST dataset, an image dataset for handwritten digit recognition). LeNet is small and easy to understand yet large enough to provide interesting results. Furthermore, the combination of LeNet + MNIST is able to run easily, making it easy for beginners to take their first step in Deep Learning and Convolutional Neural Networks.

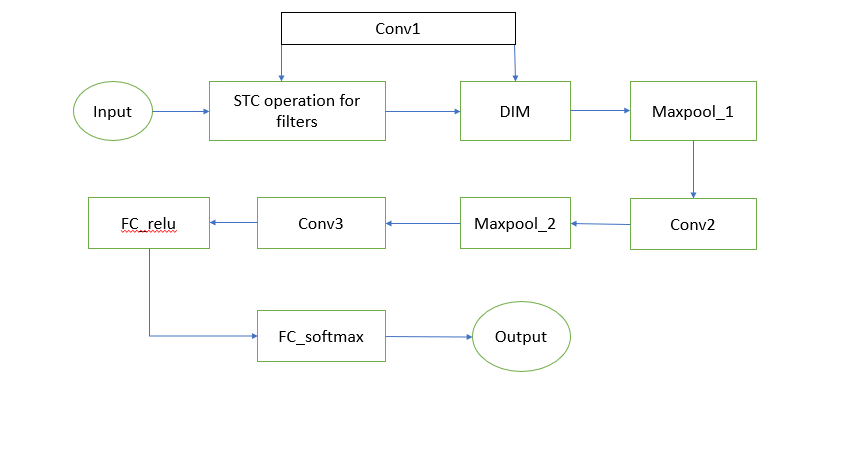
The LeNet architecture consists of the following layers:



First, we have to initialize the weight matrix and then we pass the weight matrix into the function of Sparse ternary connect then the resulting weight matrix is sparse ternarized in which most of the kernel (40 %) are filled with zeros. And the floating point integers are converted to +1 or -1(if kernel value <0 then it will be converted to -1 and if kernel value>0 then it will be converted to +1).

   Second operation or method is used to computing sparse activation function similarly to weight matrix. So, in this way we obtain convolutional neural networks by zeroing a large portion of both activations and weights. In this work we obtain over 60% (about) of the activations and weights are zero in many CONV layers of LeNet.

**Flowchart of our project implementation:**

****

**Note: Conv2 and Conv3 are the same as Conv1 and for Convolutional we are using relu activation.**

So, the methods are -

**A. Sparse ternary connect-**

In this work, we develop a weight representation method not only to highly compress the network but also to implement it very efficiently in real-time. The proposed algorithm trains the network so that the weights are represented using a structured sparse ternary format. This format allows +1 or -1 only at specified locations, while most of the values are pruned to zero.

    In the original TC, weights can be -1, 0 and 1 and an indeﬁnite number of weights will be assigned to zero. If the weights are zero, the corresponding signals need not to be accumulated. However, the original ternarized method cannot directly relate to area reduction. So, we propose STC, an improved method to explore the potential of TC and further reduce the utilization of ﬂoating adders. To do so, a parameter ρ (0 ≤ ρ ≤ 1) is introduced, and

N = |W|∗ρ weights are  set  to  zero  where  |W|  is  the  size  of kernel weights. Figure A.b illustrates the difference between TC and STC conversions. In the original TC, considering the worst case, there is no zero in a converted matrix, therefore 8 adders are necessary to accumulate the 9 element-wise product. As for the proposed STC with ρ =0 .5, there are at least 4 zero elements existing in the matrix, only 4 adders are needed for the accumulation.

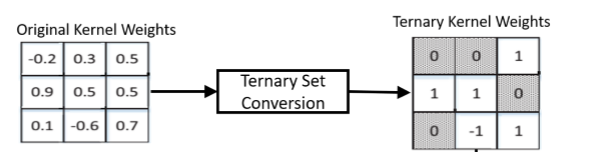


                                       Fig A.b

The computation ﬂow of sparse ternarize (W, N) function can be summarized as following 3 steps:

Step 1.  Given a customized sparsity parameter ρ, calculate the target number of weights N which need to be zeroed.

Algorithm 1

STC in typical convolutional layer l, which has n input channels and m output channels. C is the cost function and the function clip(w) clips weights into [−1,1].

Steps to Calculate Sparse ternarized weight matrix:

 Require: kernels W, bias b in this layer. Layer input x, learning rate η, customized sparsity parameter ρ.

Step 1.A.  it is to ensure that weight(W) and bias(b) are updated.

B. for forward propagation

   Calculate N using (using W and ρ)

   Calculate sparse ternarized weight

C. Compute convolution output y knowing x, Ws, t and b

D. for backward propagation

   Compute error

   Update weight(W) and bias(b)

Step 2.  Get the positions of N weights closest to zero in the ﬁlter matrix.

Step 3.  Obtain the sparse kernel by setting these N weights to be zero and making weights ternary.

Note that positions of zero depends on a kernel, so when constructing accumulation module, a selector is necessary to select non-zero parts

So, in this way we compute the weight matrix in the form of Sparse ternarized.

Now, the second approach is Dual indexing Module.

**B. Dual Indexing Module (DIM)-**

In this work, a Dual Indexing Module (DIM) is proposed to efﬁciently handle the alignment issue where activations and weights are both kept in compressed-sparse format.  Sparsity is widely observed in convolutional neural networks by zeroing a large portion of both activations and weights without impairing the result. By keeping the data in a compressed-sparse format.  Exploiting sparsity in CNN is a promising way to reduce the data. It has been shown that there exist a signiﬁcant amount of ineffectual activations and weights where the values of these data elements are zero. The study in [11] reports that over 60% of the activations and weights are zero in many CONV layers of LeNet . The sparse data can be stored in a compressed sparse format.

Adding an indexing mechanism could facilitate the handling of the compressed-sparse format. Cnvlutin removes the zeros in activations and stores the memory offsets of non-zero data for rapid selection of matched weights. Cambricon-X adopts step indexing on weights and records the distances between each non-zero data. However, these designs only utilize one type of sparsity. None of them can handle the case when both activations and weights are kept in compressed sparse format.

This paper presents a novel Dual Indexing Module to effectively and efﬁciently handle the alignment of irregularly distributed non-zero data. The indices of activations and weights are checked in parallel and the effectual activation/weight pairs can be identiﬁed and allocated for computation.

1.  Direct Indexing on both Activations and Weights –

We choose direct indexing as the compressed-sparse format in this work mainly due to its simplicity and effective handling of sparse elements. An example of using direct indexing in a sparse neural network is presented in Fig. B.a. It is implemented with vectors of indexing bits and each bit corresponds to an activation or a weight and indicates whether the value is zero. With the direct indexing information, one can discard the zeros in a sequence of data and get a direct index with ﬁxed length Tm bits and a block of data with length Tb ≤ Tm.

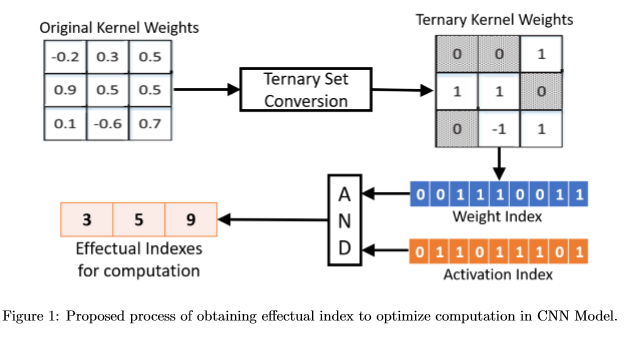


                                               Fig B.a

However, since both activations and weights are stored in the compressed-sparse format, it is not trivial to determine whether the non-zero activation and weight belong to the same pair or not. For the example in Fig. B.a, the effectual pairs that produce non-zero results are (a3, w3) and (a5, w5). Those pairs should be loaded for computation and other data should be neglected. Yet, without the indexing information, the sparse data stored in the memory have no spatial information to identify effectual pairs. To address this requirement, the proposed DIM is designed to efﬁciently pinpoint the effectual pairs of activations and weights based on the direct indexing mechanism.

Fig. 1a illustrates the detailed hardware design of DIM. The input values follow the scenario in Fig. 1a Since the bit sequence of the index value in direct indexing keeps the spatial information directly with Boolean representations, the step of DIM applies a bitwise AND operation on activation and weight indices to capture the co-activated bits. Meanwhile, DIM would accumulate the bit-sequences of the two indices respectively. These operations can generate a sequence of offsets and we need to mask out those offsets corresponding to ineffectual activations and weights. The masking process can be done by using the co-activated index to perform logical AND on the dual sets of accumulated offsets.  DIM ﬁnishes the selections with parallel multiplexers fed with masked offsets. As depicted in Fig. 1b, the design of DIM is highly parallel thus can perform the indexing task efﬁciently.

**Results and discussion**

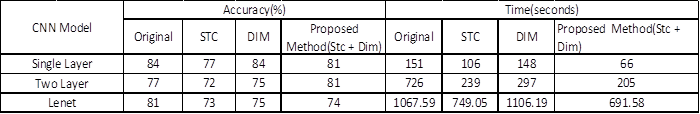
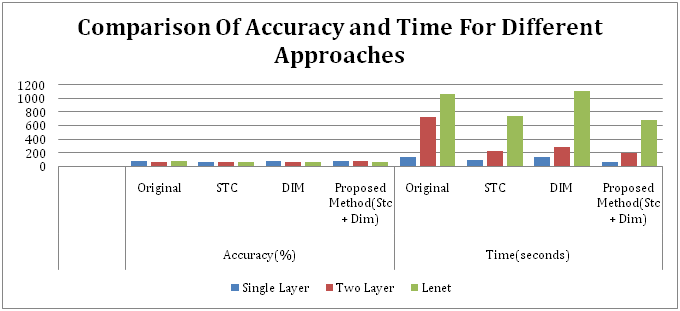
****

Fig: Comparison of Accuracy and Time for Different Approaches



In our CNN model we have used different architectures like single layer architecture, two layer architecture and Lenet-5 architecture. We have applied different optimization techniques like Sparse Ternary Connect (STC), Dual Indexing Module and combination of both i.e. (STC +DIM) approaches. Above Fig shows the comparison between time taken and accuracy obtained for original, STC, DIM and our proposed method on MNIST dataset. Use of STC in CNN reduces the number of calculations which results in time reduction. Accuracy may decrease in some amount when we use STC because we use ternarized weights in STC. We can increase accuracy of our CNN model by using DIM technique. But from the above table we can easily see that, when we use STC and DIM together it will take less time compared to other methods without much loss in accuracy. If we consider single layer and two layer architecture, our proposed method takes less than half the time taken by original CNN. We have got these results just by applying optimization techniques only for forward propagation of CNN. In future, for better results we can further extend this method to backward propagation also.

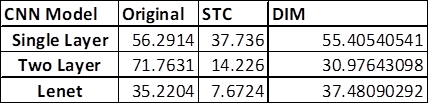
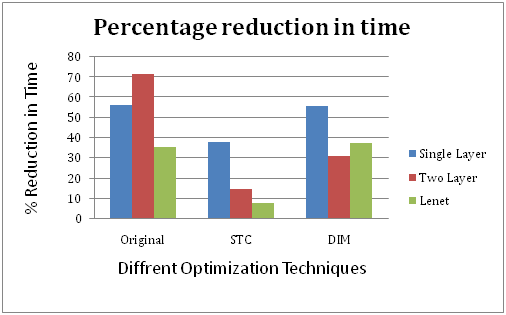


                   Fig:  Percentage reduction in Time



From the above graph we can easily observe that for original CNN and STC we have reduced much time for Lenet-5 as compared to single layer and two layer architectures. Above results show that Lenet-5 is most efficient architecture CNN when we use it with the STC method.

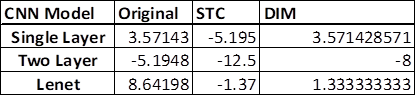
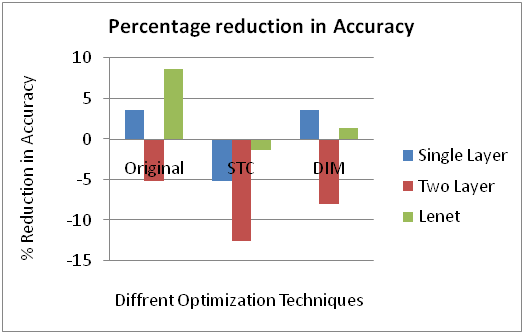


              Fig: Percentage Reduction in Accuracy



From the above graph we can observe that STC method gives poor accuracy for every architecture as compared to the original CNN. The DIM method is giving the same accuracy as the original one. Two layer architecture gives worst accuracy for each approach but Lenet-5 is giving good accuracy for each approach.

    From above observations, we can conclude that use of Lenet-5 architecture for our CNN model not only reduces computational time but also improves accuracy on the dataset. When we use our proposed method with Lenet-5 it will enhance performance of our model.

**Conclusion and Future Scope**

In recent years Convolution neural network (CNN) has been advancing rapidly and it is used widely for many computer science domains, such as image recognition, gender recognition, etc. It has accomplished astonishing achievements across a variety of domains, including medical research, and an increasing interest has emerged in radiology. In the image recognition field, in 2015, the recognition accuracy of ResNet surpassed human accuracy. The outstanding performance makes CNN more and more popular in the artificial intelligence applications.

Although deep learning has become an important method to solve a variety of complex tasks, it has its limitations. Having knowledge in key concepts and advantages of CNN as well as limitations of deep learning is important in order to make full use of it in various fields like radiology research.

Most of the important accomplishments of deep learning are actually based on very large amounts of data. In medical research, Well-annotated large medical datasets are needed. Unfortunately, building such datasets in medicine is costly and demands an enormous workload by experts. The goal of large medical datasets is to enhance generalizability and minimize overfitting, as discussed previously. In addition, dedicated medical pretrained networks can probably be proposed once such datasets become available, which may encourage the development of deep learning research on medical imaging, though whether transfer learning with such networks improves the performance in the medical field compared to that with ImageNet pretrained models is not clear and remains an area of further investigation.

**Acknowledgement**

We want to give especial thanks to our respected mentor Dr. Vijaypal Singh Rathod for helping us in all ways to complete our project. We are highly

Pleased for giving us such an opportunity.

Finally, thanks to our group members for their co-operation.

**References**

[1] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, “Neural networks with few multiplications,” arXivpreprint arXiv:1510.03009, 2015.

[2] D. Kim, J. Ahn, and S. Yoo, “A novel zero weight/activation-aware hardware architecture of convolutional neural network,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. IEEE, 2017, pp. 1462–1467.

[3] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, “Design space exploration of fpga-based deep convolutional neural networks,” in 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2016, pp. 575–580.

[4] C.-Y. Lin and B.-C. Lai, “Supporting compressed-sparse activations and weights on a simd-like accelerator for sparse convolutional neural networks,” in 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2018, pp. 105–110.

[5] C. Jin, H. Sun, and S. Kimura, “Sparse ternary connect: Convolutional neural networks using ternarized weights with enhanced sparsity,” in 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2018, pp. 190–195.