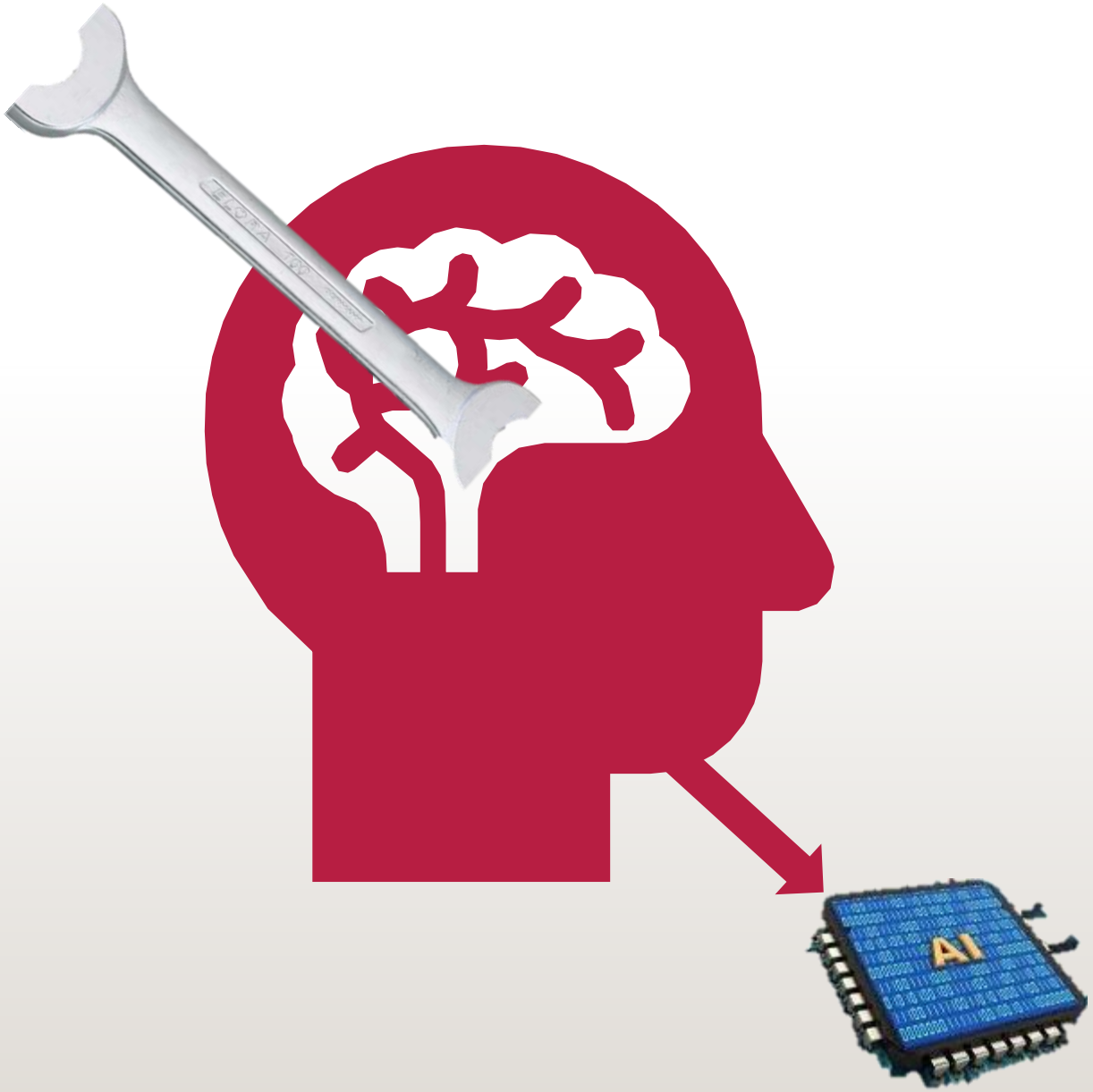


E0 270 : MACHINE LEARNING

COURSE PROJECT 2020

9 June 2020

OPTIMIZING CONVOLUTIONAL NEURAL NETWORKS

BY,

V. NAVEEN CHANDER

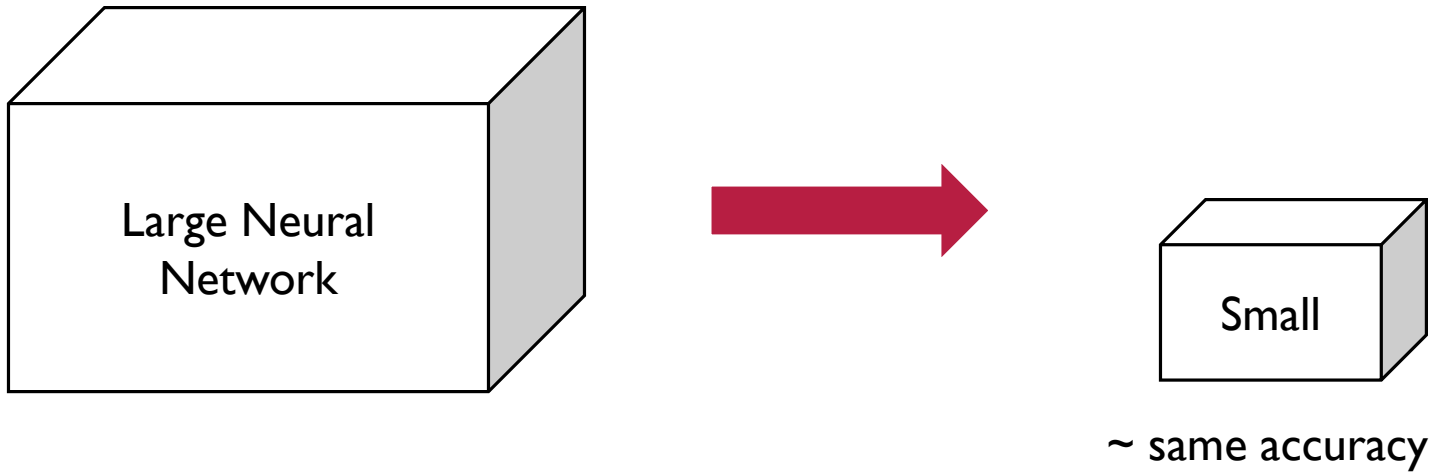
TEJAS OTURKAR,

MTECH , I ST YEAR

INDIAN INSTITUTE OF SCIENCE

GOAL : NETWORK COMPRESSION

- The goal is to reduce the size of Neural Networks without compromising on the accuracy.



OVERVIEW AND RESPONSIBILITY

Study and identify a Neural Pruning Technique	Naveen
Implement state of the art Pruning Technique on CNN and MLP	Naveen
Performance Evaluation and propose improvement in the pruning technique	Tejas
Completing the Optimization Flow by incorporating hardware optimization techniques	Tejas

WHY OPTIMIZE CNN?

- Deep Learning's success has led to rising use of AR, facial recognition, voice assistants on mobile phones, Self Driven Cars and other embedded devices.
- Emergence of IoT with edge-AI
- CNNs are heavily Over-parameterized -VGG16 has 138M parameters , Alexnet has 61M parameters

Phones


Drones


Robots


Glasses

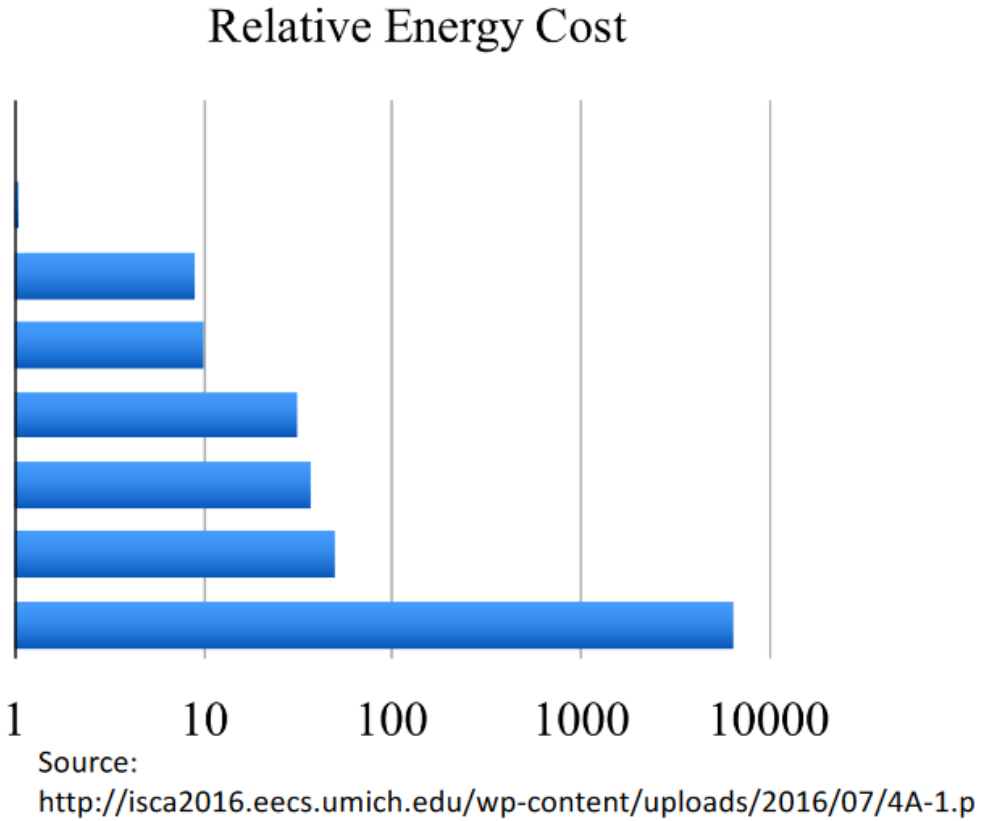

Self Driving Cars

Battery Constrained!

Source:
http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.pdf

WHERE TO ATTACK?

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	640	6400



Source:
http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.p

Say, if a Neural Network has 1 billion parameters and is running at 20 Hz, its power consumption for fetching these weights from the DRAM memory would be :
20 Hz * 1G * 640 pJ = 12.8 W!
Well beyond the power capacity of Mobile and IoT devices

Clearly, in order to speed up and miniaturize CNNs , the problem of memory storage has to be tackled.

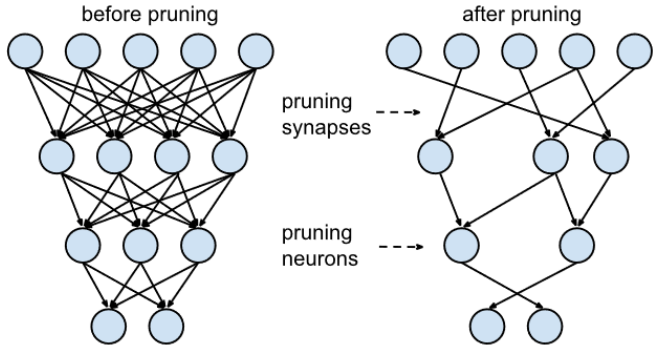
TECHNIQUES FOR EFFICIENT INFERENCE

- **Pruning:** Sparsifying neural network by zeroing out the non-significant weights
- **Weight Sharing:** Clubbing weights have close by magnitudes to a single value
- **Low Rank Factorization:** Expressing the weight matrix as a sum of sparse matrices
- **Quantization**: Converting weights to fixed-point numbers, reduce the number of different weights
- **Huffman Coding:** Assigning lesser number of bits to frequently occurring weights.

PRUNING

Definitions [Blalock, 2020]

- **Network Architecture** is a family of functions $f(x; \cdot)$ that consists of:
  - Configuration of network's parameters
  - Sets of operations to produce outputs from inputs such as convolutions, activations, pooling etc.,
  - Ex: AlexNet, VGG Net
- **Neural Network Model** is a particular parameterization of the architecture, i.e., $f(x; W)$ for specific parameters W.
- **Neural Network Pruning** entails taking an input model as $f(x; W)$ and producing a new model $f(x; M \odot W')$.
  - Here, W' is a set of parameters that may be different from W,
  - $M \in \{0,1\}^{|W'|}$ is a binary mask that fixes certain parameters to 0
  - $\odot$ is element-wise product operator



Han, Song, et al. "Learning both weights and connections for efficient neural network." NeurIPS. 2015

Algorithm 1 Pruning and Fine-Tuning

Input: N, the number of iterations of pruning, and X, the dataset on which to train and fine-tune

1: $W \leftarrow initialize()$
2: $W \leftarrow trainToConvergence(f(X; W))$
3: $M \leftarrow 1^{|W|}$
4: **for** i in 1 to N **do**
5: $\quad M \leftarrow prune(M, score(W))$
6: $\quad W \leftarrow fineTune(f(X; M \odot W))$
7: **end for**
8: **return** M, W

High Level Algorithm

- Nearly all the algorithms are derived from Algorithm 1 [Song Han et.al ,2015]
- A score is issued to each parameter or structural element after training.
- Pruning reduces the accuracy of the network, so it is trained further (known as fine tuning) to recover.
- The process of pruning and fine-tuning is iterated several times, gradually reducing the network's size.

Unstructured Pruning

- Prune weights based on their magnitude

- Techniques
  - Optimal Brain Surgeon [Stork et.al, 1993]
  - Iterative Pruning Training [Song Han et.al., 2015]
  - Weight Decay [Hanson, 1989]
  - L0 Regularization [Louizos et.al , 2018]
  - Optimal Brain Damage [Tartaglione et.al, Le Cun 1990]
  - Data Free Parameter Pruning [Babu et al., 2015]

- Benefits
  - Can achieve high compression ratios [upto 90% in SqueezeNet 2015]
  - Can achieve Regularization in some cases

- Shortcomings
  - Hyperparameters with weakly grounded Heuristics like Layer-wise pruning threshold[Dong 2017], regulariser etc.,
  - Disrupts the regularity, especially on CNNs since, the pruned kernels might be of irregular sizes. Modern Hardware can no longer exploit regularities in CNNs, thus imposing requirements for special hardware or sparse BLAS libraries.[Han et.al 2016]
  - Convergence might not be guaranteed [Lee, 2019]

Structured Pruning

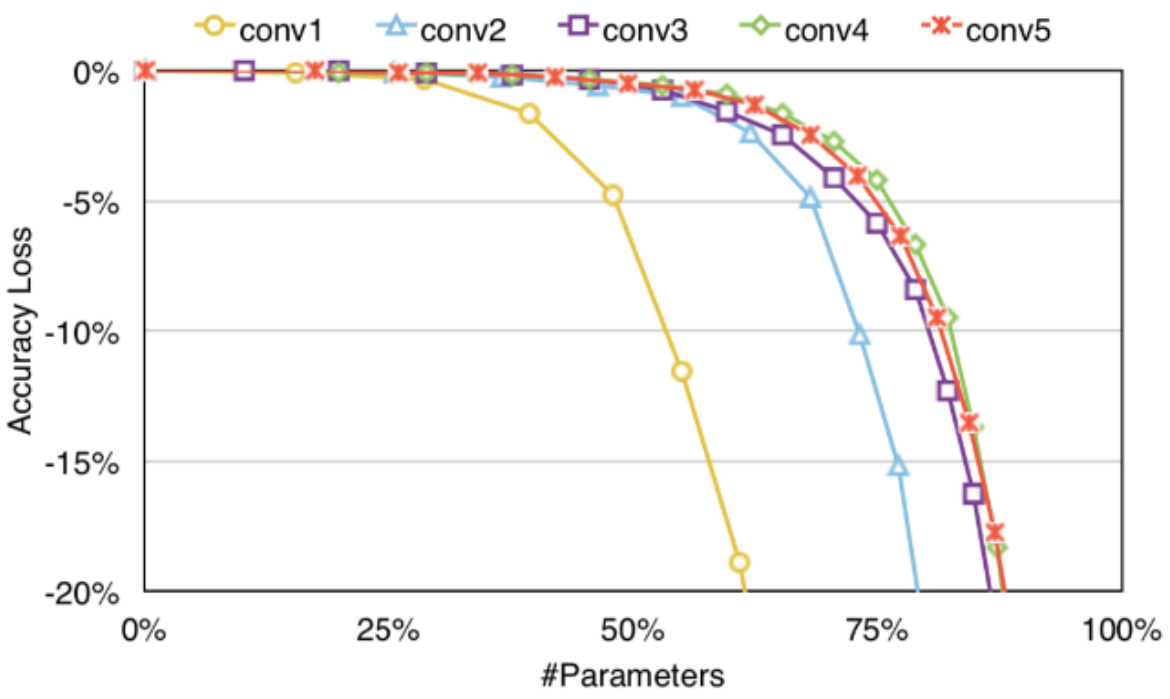
- Prune entire neuron, filter or channel to exploit hardware and software optimized for dense computation [Blalock, 2020]

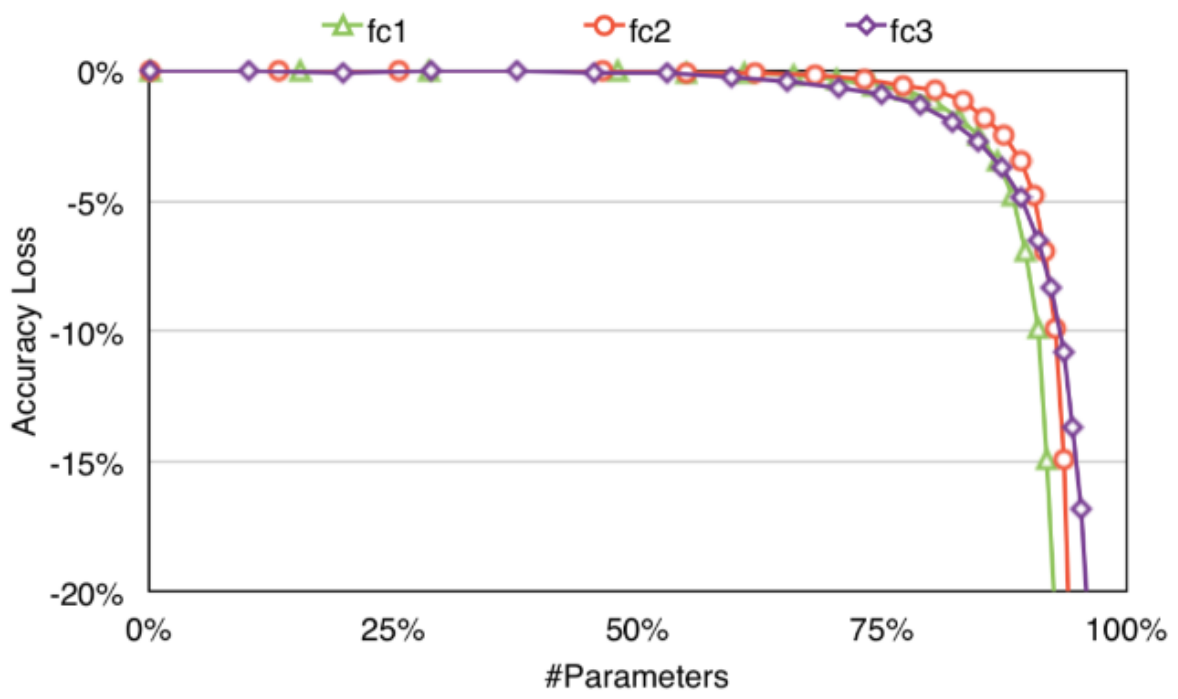
- Techniques
  - Iterative Thresholding and Shrinkage Algorithms(ISTA)
    - ℓ_1 norm based layer wise filter pruning [Li et.al, 2017]
    - Taylor expansion based pruning criteria [Molchanov 2019]
    - FLOPS regularized pruning [Molchanov et.al, 2019]
  - Self Adaptive Network Pruning [Chen et.al , 2019]
  - Neural Architectural Search
    - Transformable Architectural Search[Dong and Yang, 2019]
    - Network Slimming [Liu et.al 2017]
  - ThiNet [Luo et. al, 2017]
  - Importance Estimation in Neural Network Pruning [Molchanov et. al, 2019]

- Benefits
  - Effective for CNNs as regularity of the network is maintained after pruning.
  - Less dependence on hyper-parameters.
  - More model agonistic

- Shortcomings
  - Pruning is restrictive . Hence, the compression ratios are not as high as unstructured methods.

PERFORMANCE OF UNSTRUCTURED PRUNING FOR CONV LAYERS AND FC LAYERS





Source: Han et.al 2015.

PRUNING ALGORITHM

Based on Paper

Importance Estimation for Neural Network Pruning

[Molchanov et. al, 2019]

BUILDING THE CASE FOR IMPORTANCE ESTIMATION FOR PRUNING NEURAL NETWORKS

- Given neural network parameters $\boldsymbol{W} = \{w_0, w_1, \dots, w_M\}$ and a dataset $D = (x_0, y_0), (x_1, y_1), \dots, (x_K, y_K)$ composed of inputs (x_i) and outputs (y_i) pairs. The task of training is to minimize the error E by solving:

$$\min_{\boldsymbol{W}} E(D, \boldsymbol{W}) = \min_{\boldsymbol{W}} E(y|x, \boldsymbol{W}) \; (1)$$

- In case of Pruning we can include a sparsification term in the cost function to minimize the size of the model:

$$\min_{\boldsymbol{W}} E(D, \boldsymbol{W}) + \lambda \|\boldsymbol{W}\|_{\boldsymbol{0}} \; (2)$$

where, λ is the scaling coefficient and $\|\cdot\|_0$ is the ℓ_0 norm which represents the number of non-zero elements. →Constrained Optimization with sparsity constraints.

Eq.(2) Is non-convex , NP-hard and requires Combinatorial Search for Minimization.

- **Computational Complexity of Combinatorial Search** : $\binom{n}{k}$
  where, n = Total No. of Filters (n = 4096 for VGGNET) and k = No. of Filters to prune.
- Therefore, Combinatorial Search is not a viable option.

BUILDING THE CASE FOR IMPORTANCE ESTIMATION FOR PRUNING NEURAL NETWORKS

- **Importance Parameter**
  - Under an i.i.d assumption, define *Importance* ($\mathcal{I}_m$) as the squared change in loss induced by removing a specific filter from the network.

$$\mathcal{I}_m = (E(D, \boldsymbol{W}) - E(D, W|w_m = 0))^2 \quad \ldots\#(3)$$

  - Computing $\mathcal{I}_m$ for each parameter is computationally expensive since it requires M versions of the network, one for each removed parameter.
- Most of the methods explicitly rely on the belief that the magnitude of the weight or neuron is strongly correlated with its importance.
- [Mozer, 1988] noted that weights magnitude merely reflect the statistics of importance.
- [LeCun et al., 1989] suggested using a product of Hessian's diagonal and the squared weight as a measure of importance and demonstrated improvement over magnitude-only pruning.
- Although, the Hessian method is a textbook method which gives analytical solution, it suffers from the implicit assumption that Hessian is positive semi-definite i.e., Cost function is convex., which is rarely the case in today's DNNs wherein stochastic gradient with mini-batches are used for optimization.
- However, the proposed method :
  - Calculates importance at a very minimal which is a small fraction of the training cost.
  - Does not make use of weight magnitudes as pruning criteria
  - Hyper-parameters are minimal, therefore, methodologies are model agnostic in general

Therefore, it is considered to be a State of the art pruning method

APPROACH

- Estimate the contribution of a neuron to the final loss function and iteratively those neurons with smaller scores.
- Define *Importance* as the square change in the loss induced by removing a specific filter from the network.
- To estimate the change in the loss function analytically, Taylor Series Expansion of the Loss function is used.

PRUNING PROBLEM FORMULATION

- **Importance Parameter**
  - Under an i.i.d assumption, define *Importance* ($\mathcal{I}_m$) as the squared change in loss induced by removing a specific filter from the network.

$$\mathcal{I}_m = (E(D, \boldsymbol{W}) - E(D, W|w_m = 0))^2 \quad \ldots\#(3)$$

  - Computing $\mathcal{I}_m$ for each parameter is computationally expensive since it requires M versions of the network, one for each removed parameter.

- Taylor Series Expansion
  - Avoid evaluating M different networks by approximating $\mathcal{I}_m$ in the vicinity of **W** by its second-order Taylor Expansion as

$$\mathcal{I}_m^{(1)}(W) = (g_m w_m)^2 (4)$$

  where $g_m = \frac{\delta E}{\delta w_m}$ are elements of the gradient $\boldsymbol{g}$

  - Importance in eq.(54) can easily be computed as it is available in back-propagation.

IMPORTANCE COMPUTATION FOR A CONVOLUTIONAL FILTER

- To approximate the joint importance of a structural set of parameters W_S e.g., in Convolutional Filter, we have two alternatives. We can define it as a *group contribution:*

$$\mathcal{I}_S^{(1)} \triangleq \left(\sum_{s \epsilon \mathcal{S}} g_s w_s \right)^2 \quad \ldots (6)$$

Or alternatively, sum the importance of the individual parameter set,

$$\mathcal{I}_S^{(1)} \triangleq \sum_{s \epsilon \mathcal{S}} (g_s w_s)^2 \quad \ldots (7)$$

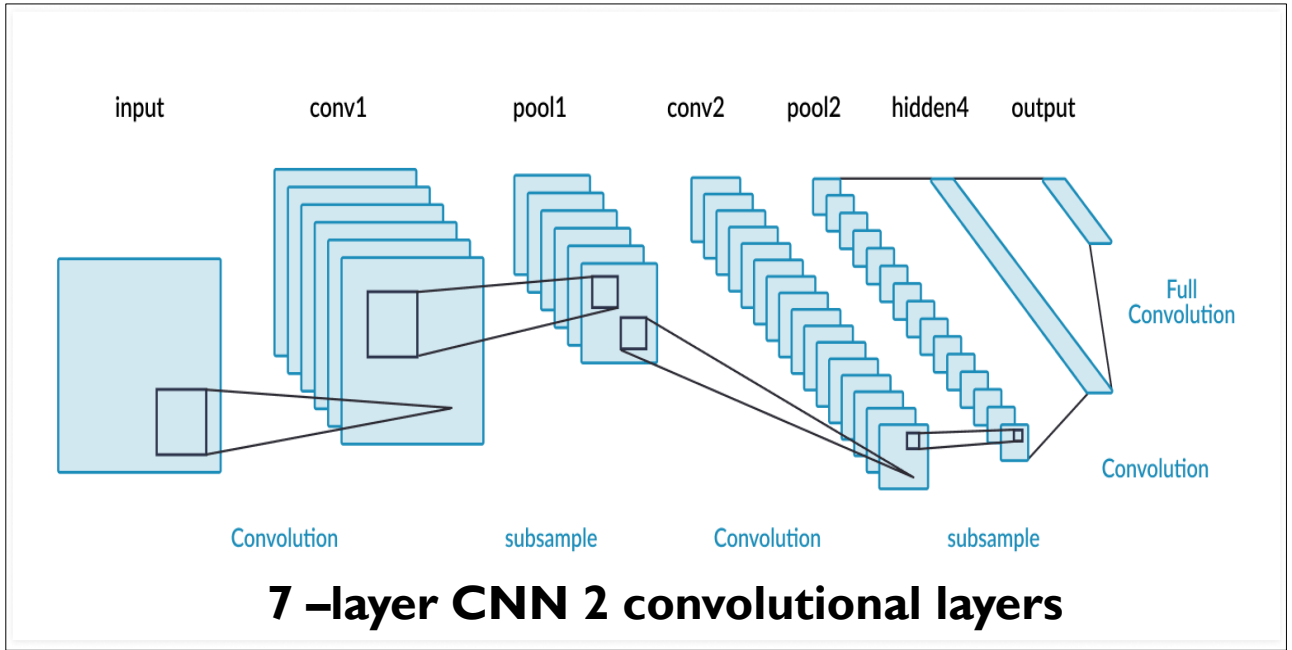
PRUNING ALGORITHM

- Train a network without any constraint
- **Iterative Pruning**
- During each epoch, the following steps are followed:
    1. For each mini-batch , compute parameter gradients and compute Importance Function using gradients averaged over the minibatch
    2. Prune N neurons with the least importance scores.
- Continue till the target number of neurons is pruned or the max. tolerable loss can no longer be achieved.

EXPERIMENTS

- We demonstrate importance pruning method on CNN and MLP topologies.
- The pruned architecture is quantized with multiplier-less optimization and synthesized on a hardware accelerator

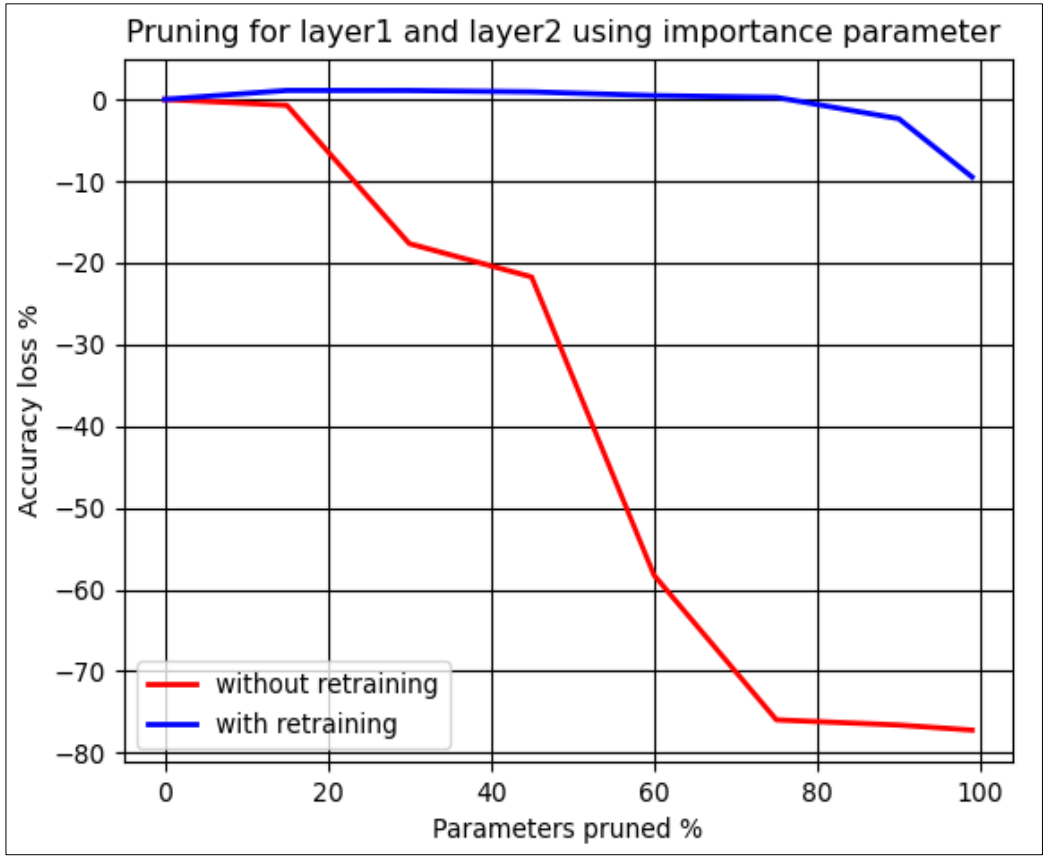
1. PRUNING CNN

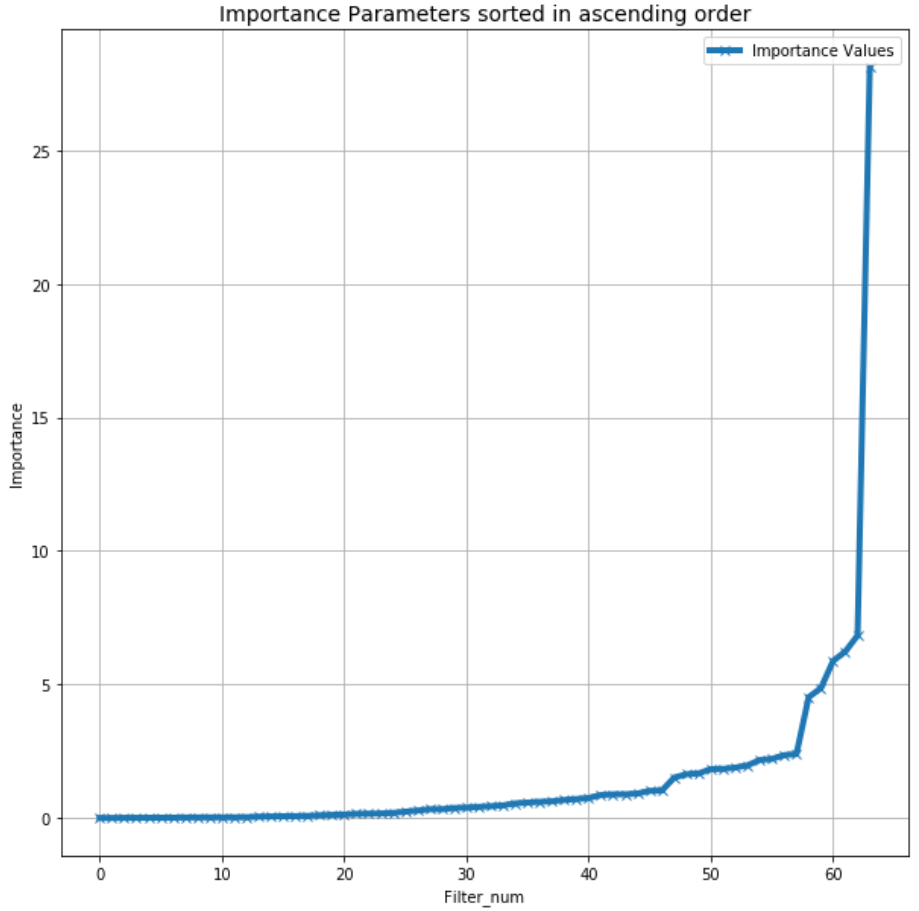


7 –layer CNN 2 convolutional layers

PRUNING CNN CONT....

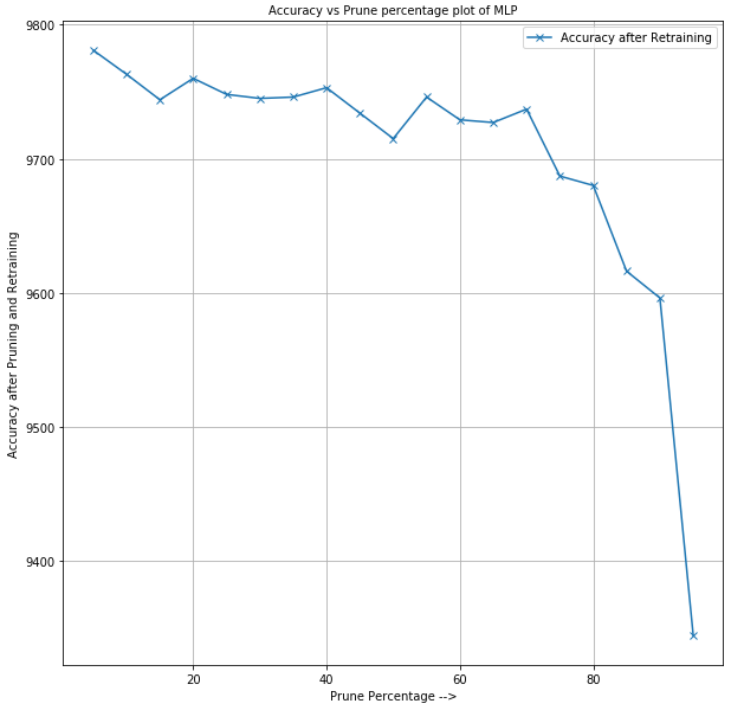
- Train the original CNN on *MNIST handwritten digits dataset* with *ADAM* optimizer to obtain 99.7% accuracy on test dataset.
- For each layer, calculate importance of all filters, sort them in ascending order of magnitude and plot.
- Iterative pruning followed by fine-tuning is carried out from 5% compression to 95% compression.

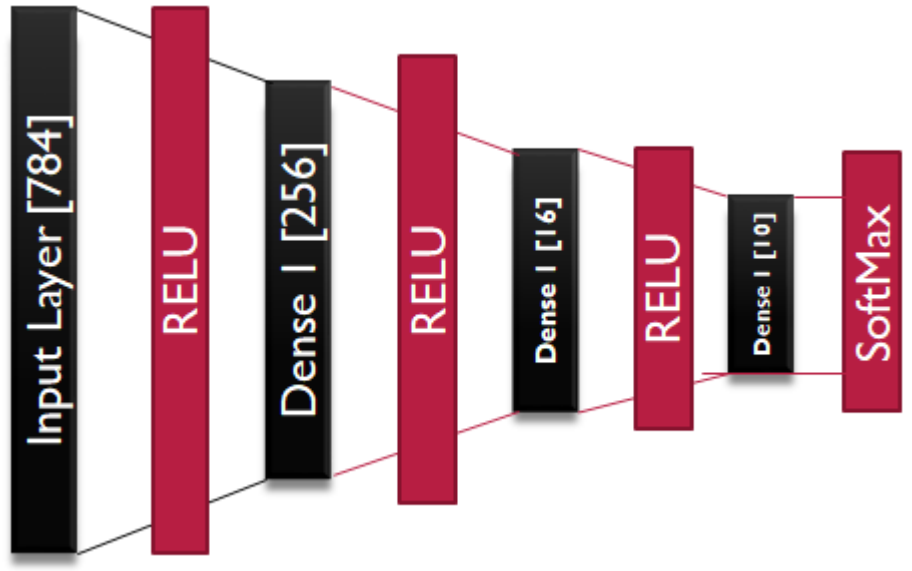




2. PRUNING MLP

- Train the original CNN on *MNIST handwritten digits dataset* with *ADAM* optimizer to obtain 97.4% accuracy on test dataset.
- Iterative pruning followed by fine-tuning is carried out from 5% compression to 95% compression.



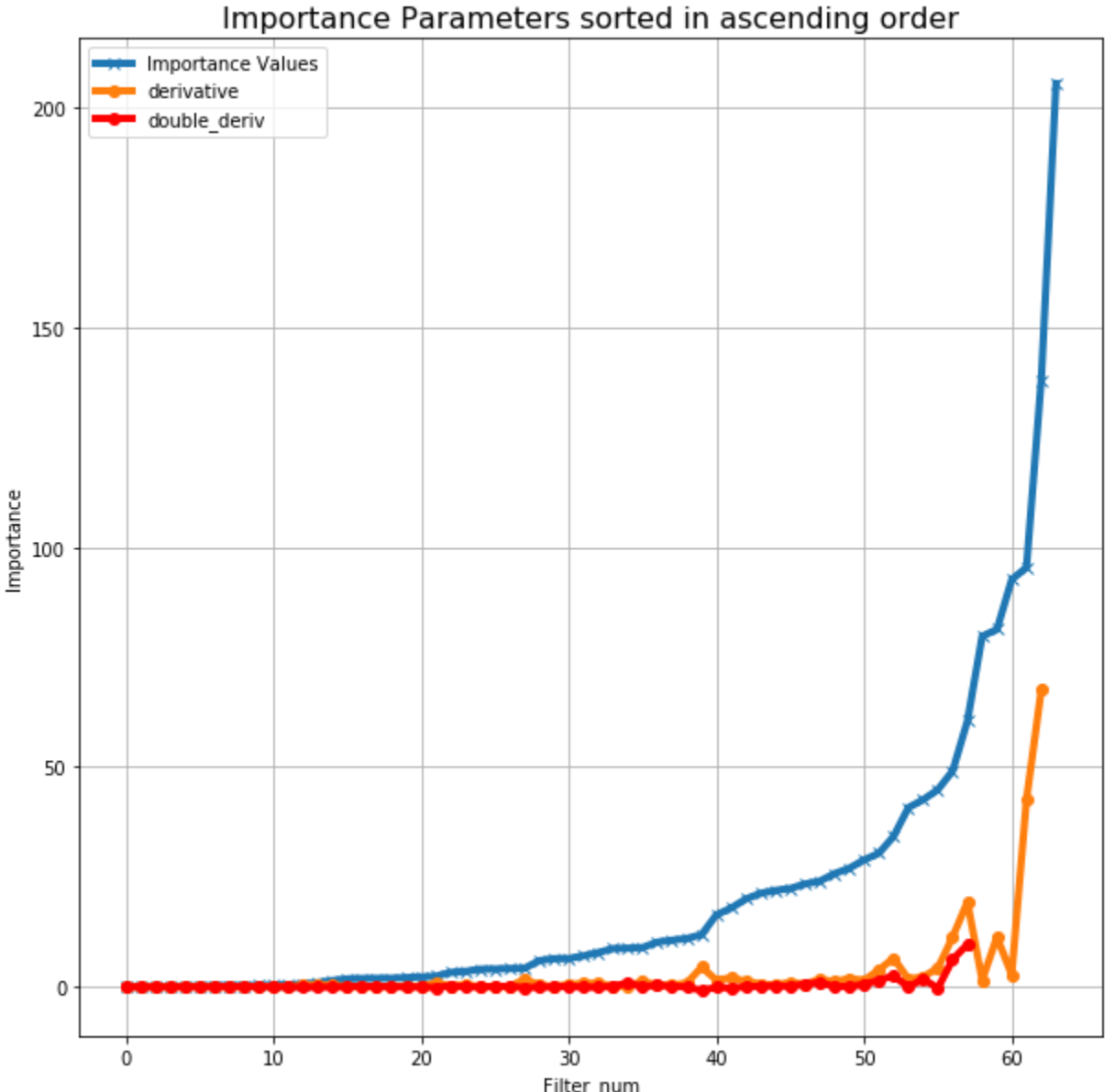


IMPROVISATION : IMPORTANCE BASED PRUNING THRESHOLD

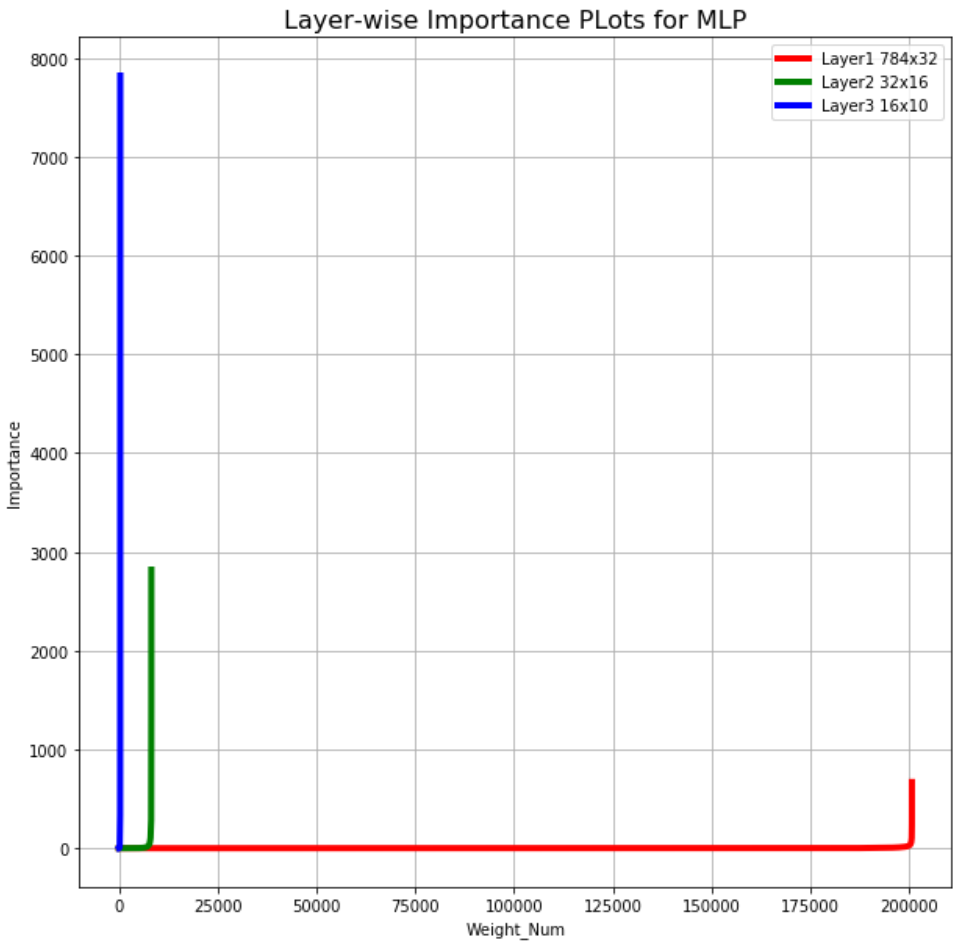
- **CNN Pruning Threshold**
- For each layer,
  - Calculate importance of parameters of each filter of a convolutional layer using formula:

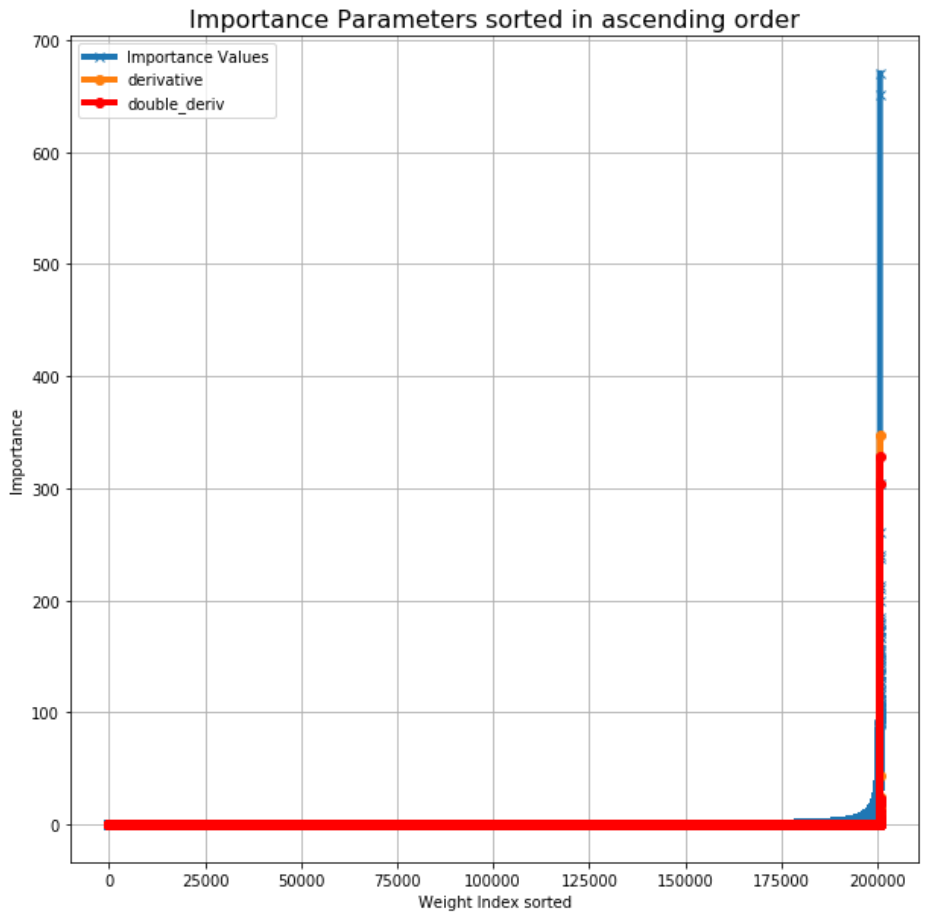
$$\mathcal{I}_p^{(1)}(W) = \left(\sum_{a=1}^{n} g_a w_a\right)^2$$

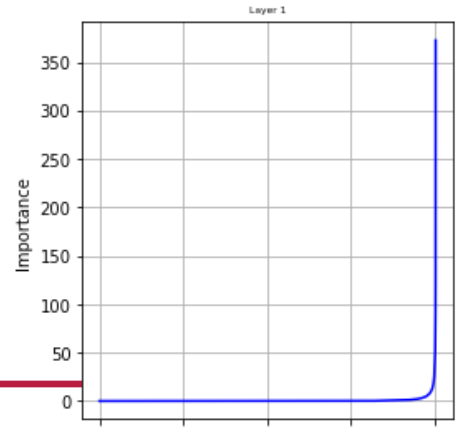
  - Sort the importance values in ascending order and plot
  - Plot double-derivative and check where the double derivative begins to rise above zero. Use a metric like *10% of the maximum double derivative value.*
  - All filters beyond this point should be retained and the rest can be safely discarded.

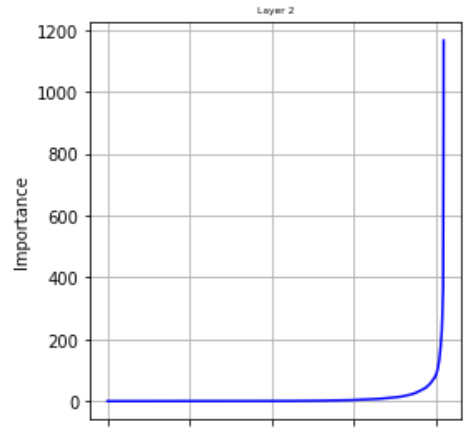


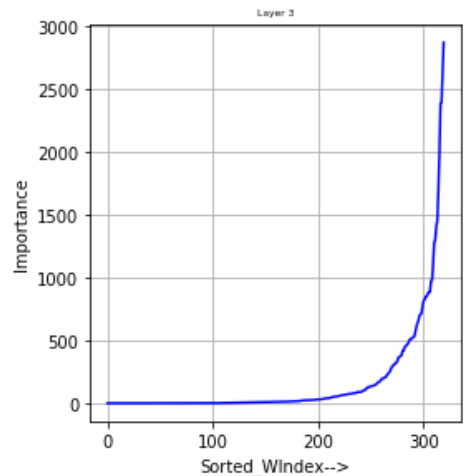
PRUNING THRESHOLD FOR MLP

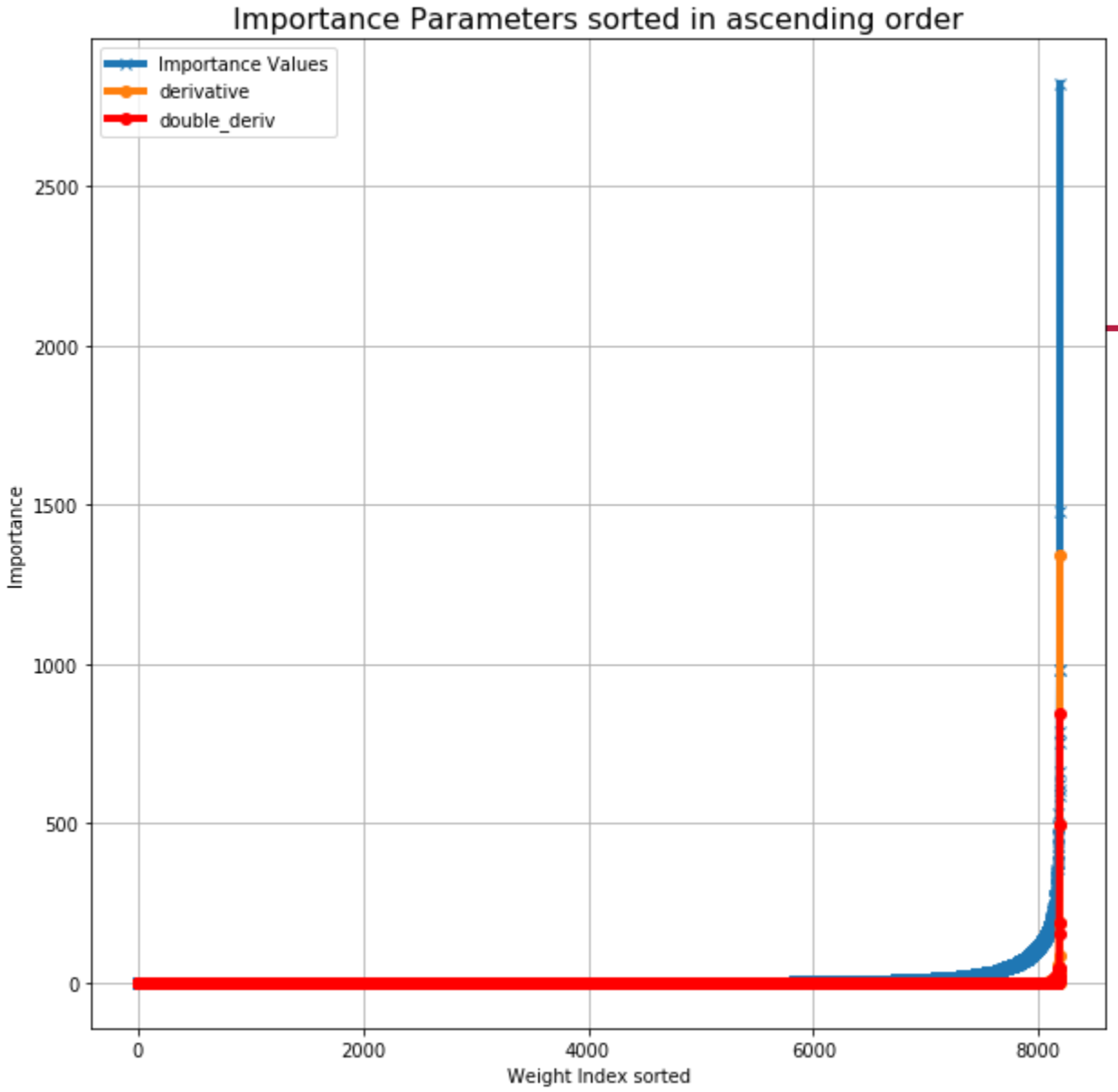




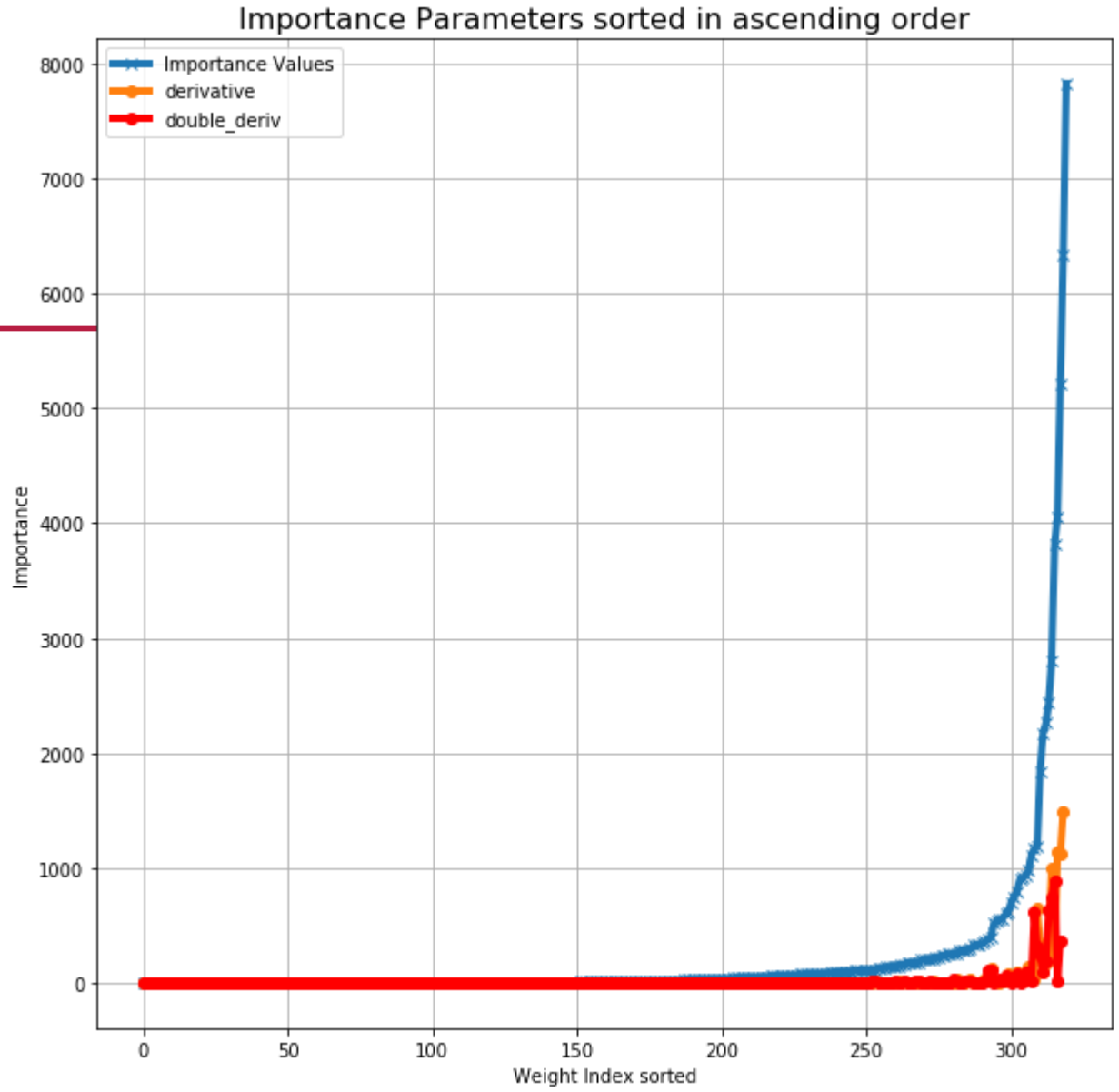








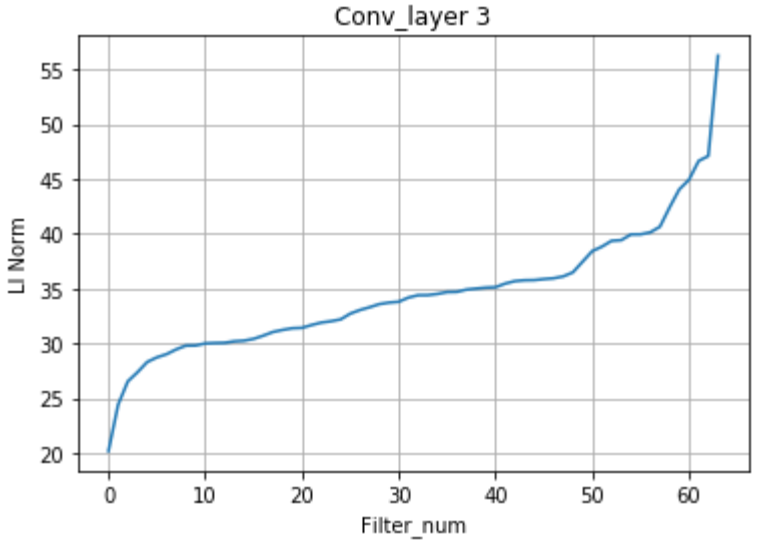
Layer 2 Importance



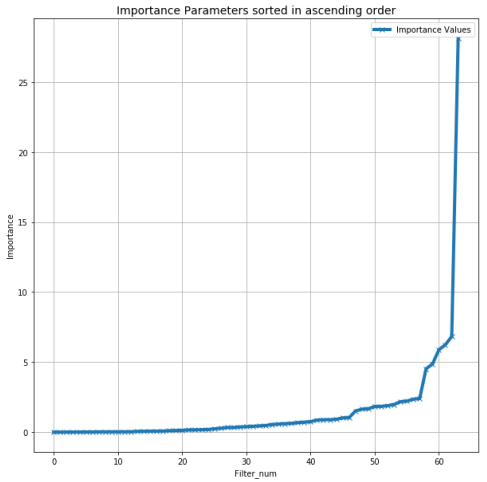
Layer 3 Importance

COMPARISON WITH OTHER PRUNING METHODS

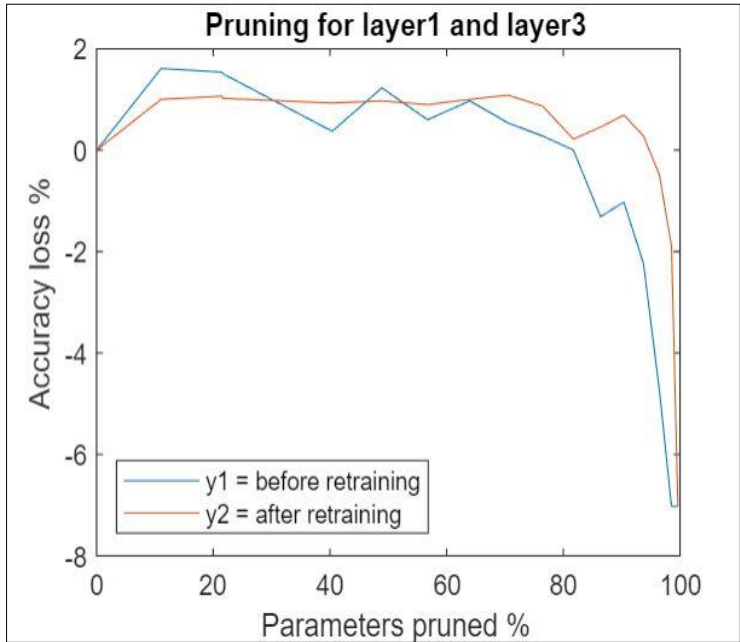
- Method **of ℓ_1 norm based structural pruning** was carried out on the 7-layer CNN shown in Figure (2) below for MNIST dataset.
- Firstly, CNN was trained on MNIST data set of handwritten numbers with 96.8% of test data accuracy. Depending on *saliency score* of the filter kernel channels were pruned in layers conv1(layer1) and conv2(layer3) after training the network. *Saliency score* is the L1 norm of filter kernel.
- $e_l^k = \sum_{i,j} |f_{i,j}^k|$
- Where, e_l^k represents norm of the kth filter kernel in the lth layer of the network.



L1 Norm Sorted



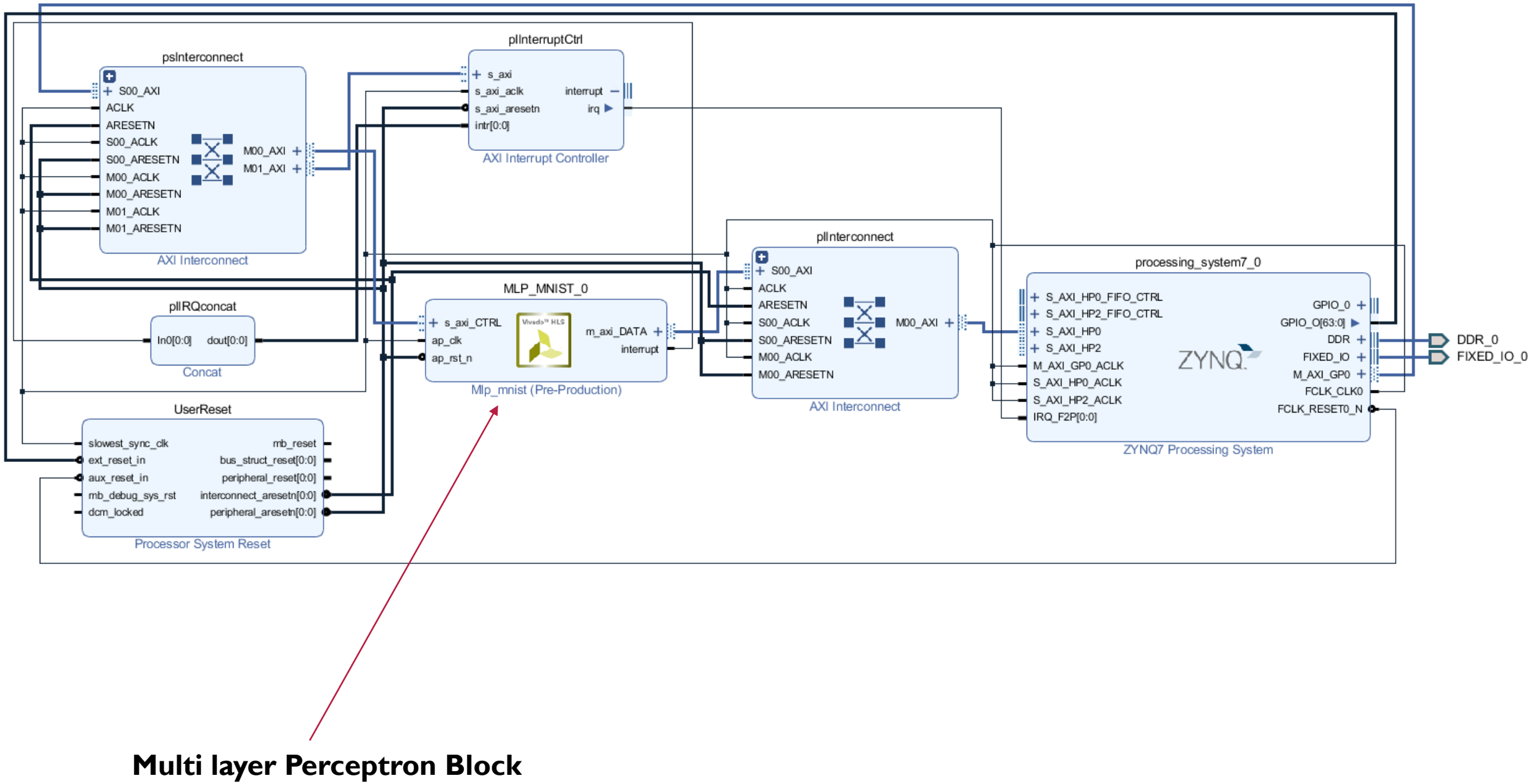
Importance Value Sorted



COMPLETING THE OPTIMIZATION FLOW

- The following steps were carried out for MLP
- Quantization – Fixed Point 16 bit
- Multiplier less Optimization – All weights quantized to powers of 2
- Hardware Implementation – Hardware Accelerator FPGA Zynq 7020 FPGA Synthesis was done and the following were the results

Parameter	Original MLP	Optimized MLP
Latency (ms)	23.06	0.52
Mem Used (KB)	9600	432



Multi layer Perceptron Block

CONCLUSION

- A unified framework for Optimizing Neural Networks has been developed

FUTURE WORK

- Validate the improvisation proposal on bigger datsets like CIFAR-10 , Image Net on larger CNNs like VGG, Inception, ResNet etc.,
- All parameters are assumed to be i.i.d. This may not be the case in reality. Importance Surfaces may required to be analyzed instead of just one layer at a time.

THANKYOU

APPENDIX : OPTIMAL BRAIN SURGEON

- Unstructured Pruning (Pruning Weights) :. Important Works:-
  1. **Optimal Brain Surgeon(1993)**
     - Analytical way of arriving at weights to be pruned.
     - Based on the idea in Wald Statistics to estimate the importance of a parameter in the model
     - Cost Function is Training Error. The weights leading to smallest change in training error are pruned.
     - **Formulation :** Consider a trained neural network with weights $\boldsymbol{w_{ij}}$. Predicted functional increase in error for a change in full weight vector $\delta \boldsymbol{w}$ is

$$\delta \boldsymbol{J} = \left(\frac{\partial J}{\partial \boldsymbol{w}}\right)^{t} . \delta \boldsymbol{w} + \frac{1}{2} \delta \boldsymbol{w}^{t} . \left(\frac{\partial^2 J}{\partial \boldsymbol{w}^2}\right) . \delta \boldsymbol{w} + \boldsymbol{O}\left(\|\delta \boldsymbol{w}\|^{3}\right) \qquad \ldots (\mathbf{1})$$

where, $\delta \boldsymbol{J} = change\ in\ cost\ function\ due\ to\ change\ in\ weights.$

     - Since, weights are obtained through back-propagation, we have : $\frac{\partial J}{\partial \boldsymbol{w}} = \mathbf{0} \ \ldots (\mathbf{2})$
     - Neglecting the higher order terms, we are left with the second term, which is a function of the Hessian of the training error.
     - The general solution for minimizing (1) given the constraint of deleting one weight can be analytically found out to be:

$$\delta \boldsymbol{w} = -\frac{\boldsymbol{w_q}}{[\boldsymbol{H_{qq}^{-1}}]} . \boldsymbol{H^{-1}} . \boldsymbol{u_q} \ldots (\mathbf{3}) \ \text{ and } \ \boldsymbol{L_q} = \frac{\mathbf{1}}{\mathbf{2}} \frac{\boldsymbol{w_q^2}}{[\boldsymbol{H_{qq}^{-1}}]} \ldots (\mathbf{4})$$

where , $\boldsymbol{u_q}$ is the unit vector along q^{th} direction in the weight space

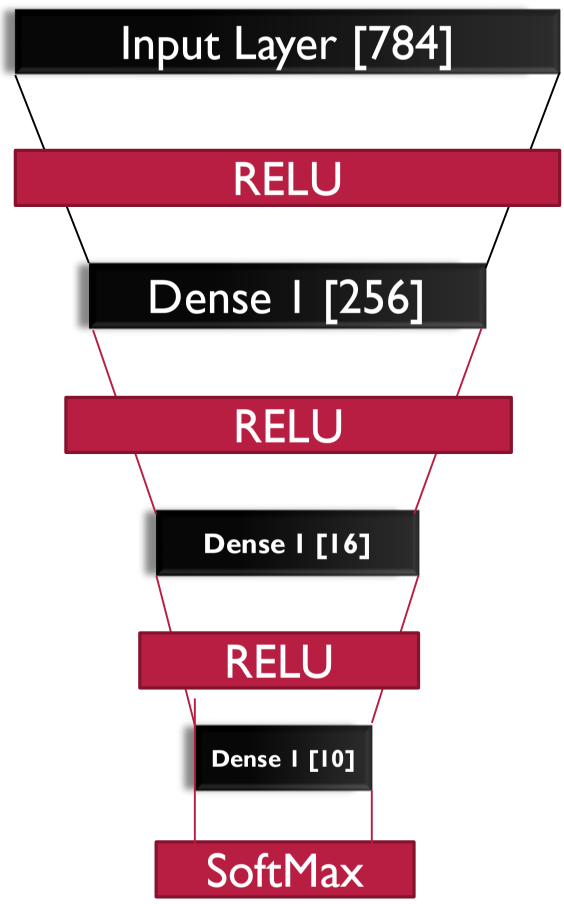
$\boldsymbol{L_q}$ is the increase in training error if weight q is pruned, a.k.a **Saliency.**

OBS : PRUNING ALGORITHM

1. Choose a Neural Network Architecture
2. Train the NN until a reasonable solution is obtained
3. Compute the second derivative for each of the weights (Hessian Matrix)
4. Compute the Saliency for each weight using eq.4.
5. Sort the parameters by saliency and delete some weights below a saliency threshold.
6. Iterate to Step 2

Drawbacks

1. Computation of Hessian is tedious. The Optimal Brain Damage technique(1989) technique assumes Hessian to be a diagonal matrix thus simplifying Step 3. Inspite of this, the overall computational complexity of OBS and OBD for DNN pruning is very HIGH.
2. Not scalable on DNNs. The assumption in eq.2 is hardly true in case of DNNs due to their complexity.


Input Layer [784]
RELU
Dense 1 [256]
RELU
Dense 1 [16]
RELU
Dense 1 [10]
SoftMax