

B. TECH. PROJECT REPORT

On

Approximate Deep Learning

BY
Samarth Anand
Vijay Babu Kumpatla



DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

Approximate Deep Learning

A PROJECT REPORT

*Submitted in fulfillment of the
requirements for the award of the degrees*

of
BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE ENGINEERING

Submitted by:
Samarth Anand
Vijay Babu Kumpatla

Guided by:
Prof. Kapil Ahuja



INDIAN INSTITUTE OF TECHNOLOGY INDORE
May, 2022

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **Approximate Deep Learning** submitted in fulfillment for the award of the degree of Bachelor of Technology in Computer Science Engineering completed under the supervision of **Prof. Kapil Ahuja, Computer Science and Engineering**, IIT Indore is an authentic work.

Further, I/we declare that I/we have not submitted this work for the award of any other degree elsewhere.

K. Vijay Babu

[Signature]
20105/22

Signature and name of the student(s) with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

KA

Professor Kapil Ahuja, CSE, 25/05/2022

Signature of BTP Guide(s) with dates and their designation

Preface

This report on “Approximate Deep Learning” is prepared under the guidance of Prof. Kapil Ahuja.

(Through this report we have tried to improve the approximation of deep learning using sampling and tried various sampling techniques, if the technique is effective.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner.)

Samarth Anand

Vijay Babu

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgements

We wish to thank Prof. Kapil Ahuja for his kind support and valuable guidance. We would also like to thank Mr Ashish Kushwah for his constant support throughout the course of this project. Lastly we would like to thank all our friends and family members for keeping us motivated and helping us as and when required.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support, this report would not have been possible.

Samarth Anand

Vijay Babu Kumpatla

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Abstract

Deep Learning is used to solve complex day to day problems. Solving of the machine learning tasks requires making use of large size DNNs. But to the sheer size and computation cost associated with them, it is very difficult to deploy these DNN models into the embedded systems. The scale of deep neural network models grows, requiring more computing and memory space. As a result, these models may be implemented on low-power devices with some approximation while maintaining network accuracy.

Many recent studies have focused on reducing the size and complexity of these DNNs using various techniques. Quantisation of the DNN model parameters is one such technique that focuses on reducing the sheer size of the model by representing the high precision parameters in some lower bit-width representation. In this thesis we will try to look upon some of the efficient quantisation techniques that does not require any retraining/fine-tuning of the model post quantisation.

We will also try to reduce the quantisation induced error by making use of Sampling Techniques.

Table Of Contents	6
1. INTRODUCTION	1
1.1 Background	2
1.2 Motivation	2
1.3 Objectives	3
2. Literature Survey and Research:	4
2.1 Survey On Quantisation Aware Training	4
2.1.1 Summary of papers read for Quantisation Aware Training	5
2.1.2 Conclusions Drawn from these papers	7
2.2 Survey On Pruning Redundant Weights	7
2.2.1 Summary of papers read for Pruning Redundant Weights	7
2.2.2 Conclusions Drawn from these papers	9
3. Recent Work:	10
3.1 Motivation Behind Align and L2L	10
3.2 log2lead Quantization Technique	10
3.2.1 Proposed Template for Quantization	11
3.3 ALigN: Layerwise Quantization of Parameters of Pre-trained Neural Networks	13
3.3.1 Proposed Template for Quantization	13
4. Proposed Methodology:	15
4.1 Linear Systematic Sampling	15
4.2 Advantages of using Linear Systematic Sampling	16
4.3 Pseudocode for Linear Systematic Sampling	17
4.4 Flowchart explaining the working of Linear Systematic Sampling	18
5. Results and Experimentation:	19
5.1 Sampling And Quantisation	19
5.2 Pruning to remove some of the redundant weights in FC layers of VGG-16	24
6. Conclusion and Future Work:	25

List Of Figures

1	Fig1. Accuracy for various bit width quantisation	6
2	Fig2. "leading one" position of weights and biases in Conv1 layer	12
3	Fig3. Proposed Template for Log2Lead Quantisation	13
4	Fig4. Example of Log2Lead Quantization	13
5	Fig5. Proposed Template for AligN Quantisation	14
6	Fig6. Example of Linear Systematic Sampling	17
7	Fig7. Pseudo Code for Linear Systematic Sampling	18
8	Fig8. Flowchart for Linear Systematic Sampling	19
9	Fig9. Top 1 Mismatch recorded for 25% Sampling Vs 100% Sampling	21
10	Fig10. Top 5 Mismatch recorded for 25% Sampling Vs100% Sampling	22

List Of Tables

1	Table1. Result for Linear Systematic Sampling applied over VGG-16 Model	19
2	Table2. Optimal no of bits selected by AlignN Quantisation for storing leading1 parameters VGG-16	22
3	Table3: Results of pruning in the FC layers of VGG-16 Model	23

1. INTRODUCTION

Deep learning is a rapidly increasing branch of machine learning that is frequently utilised for cutting-edge applications such as image classification, speech recognition, visual object identification, and object detection. DNNs are deep learning models that use many processing layers to discover data patterns across several levels of a neural network. Complex machine learning applications need large DNNs with millions of processing units and learning parameters. The training and inference phases of these large-scale DNNs need a lot of computation and memory. As a result, high-performance architectures like as graphics processing units are used to train these network models (GPUs). However, due to the high processing requirements, the implementation of these Deep Neural Networks onto some embedded systems and end-user devices is almost impossible. DNNs are used in a variety of applications currently. The following are a handful of them.

Deep learning-based decision-making frameworks in self-driving vehicles acquire streams of data from a range of onboard sources, such as digital cameras, radars, and other various sensors. These streams of data are then used to train and infer the deep neural networks. Furthermore, photo captioning involves several stages such as including image encoding. Deep neural networks such as “AlexNet”, “VGGNet”, “ResNet”, “GoogLeNet”, and “Inception” networks are extensively utilised for image encoding. To encode these models, a lot of memory and calculations are required. In the healthcare industry, deep neural networks are utilised for cancer diagnosis, picture segmentation, and identification.

In this thesis, we will look at some of the strategies to reduce computations and memory requirements for the implementation of Deep Neural Network models on some of the low-computation devices.

1.1 Background

Large DNNs are used in machine learning to solve complicated classification tasks. Deep neural networks employ feedforward and backpropagation in their training and inference processes. These DNNs have a greater number of training parameters due to their complicated structure. For training neural networks, high-performance parallel architectures such as GPUs are employed. For high network accuracy, these topologies employ single-precision floating-point parameters. But to deal with so many network characteristics on some low-performance devices, however, necessitates certain approximation approaches. A slew of recent papers have proposed several low-bit-width quantization approaches and hardware accelerators to help trained DNN models save memory and compute time. Sparsity in networks, dynamic fixed point and logarithmic quantization techniques, sampling-based methodology, and clustering-based technique are all used to approximate deep neural networks. To reduce memory footprint and computing cost, Google cloud tensor processing units (TPUs) adopt BFloat16, a 16-bit numeric format with 7 bits for storing fraction portion. The system may construct more complicated deep learning models with a minor compromise in number precision by making the use of 16-bit format in place of 32-bit single-precision floating-point format. Furthermore, only a few studies suggest DNN computation with only 8 bits or even less. Some of the proposed solutions include retraining or fine-tuning DNN to compensate for low bit-width parameter accuracy loss.

1.2 Motivation

DNNs can withstand small errors. In DNN, a modest change in parameter precision has no discernible effect on network accuracy. Taking advantage of this aspect, recent research has demonstrated that employing various network approximation approaches, large-scale deep neural networks may be constructed with low computing and storage requirements. However, in order to reduce the inaccuracy caused by approximation, these approximation strategies need computationally expensive network retraining. As a result, there is a need to design approximate procedures that do not require retraining. Iterative quantization approaches for obtaining lower bit-width parameters of a network have been developed by a few researchers. These iterative methods might take a long time. We present a quantization-based approach for pre-trained deep neural networks to avoid the retraining and iteration process. Some academics have developed compression

methods based on layerwise network analysis. We also offer the parameter representation approach, which examines each DNN layer and further lowers parameter quantization error. Multiplication is the most computationally intensive operation in deep neural networks. Few studies use bit-shift and addition procedures to avoid multiplication operations in DNNs. For efficient DNN inference, our suggested approach additionally employs bit-shift and addition. Furthermore, for various machine learning applications, several works have created DNN accelerators based on GPU, FPGA, and ASIC hardware. A next goal for our research is hardware accelerators employing our proposed parameter approach. Overall, the goal of this thesis is to address a variety of difficulties related to quantization-based approximation for effective DNN training and inference.

1.3 Objectives

We hope to accomplish the following goals with this thesis:

- (i) Implement a “quantization-based parameter representation” approach for pre-trained deep neural networks that preserves parameter accuracy after quantization.
- (ii) To make the approach scalable for different state-of-the-art DNNs in terms of bit-width.
- (iii) Implement a quantization approach that can give multiple quantization setups for distinct DNN layers.
- (iv) To implement a quantization approach for DNN inference that allows for multiplier-less arithmetic.
- (v) To explore whether we can improve the accuracy of the resultant model by using sampling techniques on top of the quantization.

2. Literature Survey and Research:

2.1 Survey On Quantisation Aware Training

Why should we use QAT?:

Scalar quantization, for example, replaces the weights representation of a trained network from floating point to some lower-precision bit-width representation, such as fixed-width integers .

Applying this technique, we can achieve a high compression rate while also speeding up inference on the underlying hardware. But the inaccuracies introduced by the use of these approximations accrue in the forward pass computations, resulting in a considerable loss in performance.

Furthermore, while the post-training quantization technique works well for big models with high representational capacity, it causes severe accuracy losses in tiny models.

For different output channels, there are significant weight range fluctuations (more than 100) (“mandates that all channels of the same layer be quantized to the same resolution, which causes weights in channels with smaller ranges to have much higher relative error”)

One of the ways that can be followed to rectify the drifting effect would be to directly quantize the network during the training phase itself. This would raise the following two challenges:

- 1.) The discretization operators, for starters, will be having a null/zero gradient, which means that their derivative with respect to the input is nearly always zero.
- 2.) The second issue that frequently arises as a result of these workarounds is the inconsistency that occurs in the network's train/test functions.

Quantization Aware Training (QAT) (Jacob et al., 2018[5]) addresses these concerns by using quantisation on all the weights during the forward pass and computing the gradient with a straight through estimator (STE). This works when the STE error is modest, as with “int8 quantization”, but it is insufficient in compression regimes when the compression approximation is more severe..

The computational expense of retraining the NN model is the key downside of QAT. For low-bit precision quantization, this re-training may be required for many hundred epochs to restore accuracy. This effort in re-training is likely to be worthwhile if a quantized model will be used for a lengthy period of time and if efficiency and accuracy are particularly essential.

2.1.1 Summary of papers read for Quantisation Aware Training

In[2] the author uses the quantisation scheme on a distinct random subset of weights during each forward pass, allowing the unbiased gradients to flow through the remaining weights. Controlling the amount and structure of noise allows for the higher compression rates as well as maintaining the actual performance of the original model.

The noise function ϕ mimics the weight changes caused by the target quantization approach.

Matrix W is then replaced by the resulting noisy matrix

“ W_{noise} during the forward pass to compute a noisy output y_{noise} , i.e.,

$$W_{\text{noise}} = (\psi(bkl \mid J))kl \text{ and } y_{\text{noise}} = x * W_{\text{noise}}$$

where x is an input vector.”

During the backward pass of training, they make use of STE to replace the distorted weights W_{noise} with their non-distorted equivalents.

When J includes all the tuples of indices, their technique is identical to QAT.

“MedQ: Lossless ultra-low-bit neural network quantization for medical image segmentation”[10] introduces a new CNN quantization framework that can reduce the bitwidth of a deep neural network model (including both the parameters and the activation) to as little as 1-2 bits while retaining good performance. The authors suggest an unique adjustable quantizer that can be tuned for both internal quantization error and final segmentation loss. They also design a radial residual connection scheme and multiple training strategies to address the internal information loss caused by quantization, which significantly improves their model's performance in binary and ternary quantization, as demonstrated by recent studies.

Weight initialization[11] is one of the most important factors that affects the training process of feed-forward neural networks and the model's ultimate quality. Different approaches for examining weight matrices are presented in this article, allowing for the evaluation of initial initialization quality.

Weight initialization is one of the most important step for training of CNN models for efficient convergence of the model

Another paper read on QAT[7] uses Spike Timing Dependent Plasticity to train an SNN in an unsupervised way in this study. The training process is enhanced by applying the quantization approach to its weights at regular intervals throughout the training. According to the simulation findings, the SNN with 4-bit weights had a 0.09 percent accuracy loss. The accuracy of the behavioral simulation result with the same bit-width weight quantization dropped by 0.28 percent.

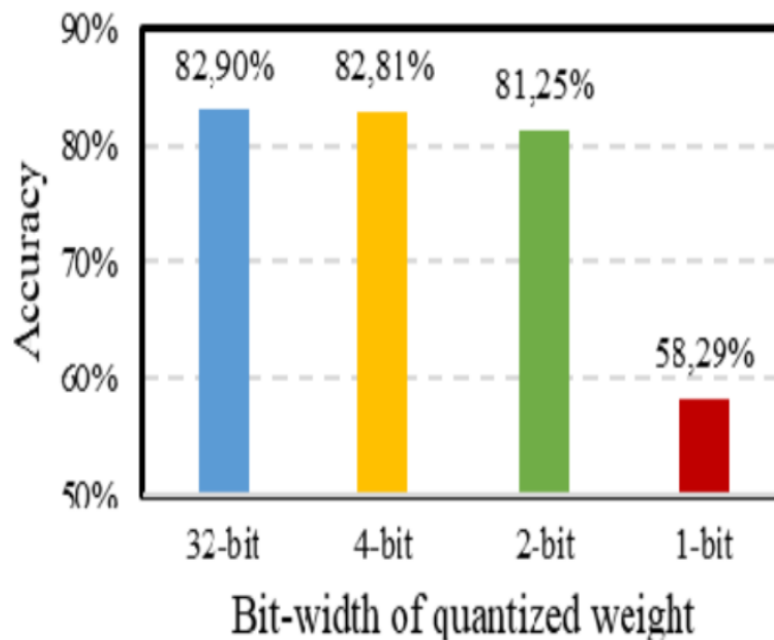


Fig1. Accuracy for various bit width quantisation[Ref. from [7]]

Dropout is another technique where we intentionally deactivate some of the neurons during the training/inference process so as to prevent the model from overfitting on some of the dataset.

In[15] they employed LSTM with dropout regularisation layers in between the word embedding layer and the LSTM layer to avoid over-fitting to the training dataset. Researchers used this strategy to reduce the weights of 20 percent of the neurons in the LSTM Layer at random.

In[14], a study was conducted to use stratified sampling strategy to sample 100 training data as in prior studies, while the remaining 479 data is used to test the model. The basic concept behind stratified sampling is to divide a variable's whole range into multiple sub-ranges and sample the data so that the probability mass in each sub-range is comparable between the sample and the population.

2.1.2 Conclusions Drawn from these papers

Quantization Aware Training seems to overcome the significant errors induced when quantising a pretrained model.

However, QAT seems to provide good results when applied on small models due to the computational cost of retraining a large model.

Hence, since we are interested in approximating large models[VGG16 / AlexNet / ResNet] for our project we decide not to move ahead with QAT as the tradeoff is not justifiable in this case

These papers present us with some state of art quantisation techniques where they go as low as utilizing 1-2 bits for quantising the weights of CNN models.

However, these techniques again require using the quantisation techniques during the training phase/retraining of the entire model itself in order to achieve desirable model accuracies.

2.2 Survey On Pruning Redundant Weights

Many of the weights in a fully-trained Neural Network are quite close to zero. This means that these weights have low to no impact on the Neural Network's output. If we imagine the Neural Network as a brain, we can see that only a tiny fraction of it is really employed to solve an issue, while the rest of it is dormant.

We travel over the CNN layers in the model using this notion, seeking for filter channels with weights that are extremely close to zero and deleting them totally. Following that, we link the various CNN layers as needed (for example, using a layer to pad an input/output tensor with zeros to acquire the correct shape as

needed) to ensure that information continues to flow appropriately through the model. As a result, the model is significantly smaller while maintaining its ability to categorise images and recognise objects.

2.2.1 Summary of papers read for Pruning Redundant Weights

Using the Ising energy of the network[16], the author presents an adaptive strategy for dropping the visible and hidden units appropriately in a deep neural network. The first findings demonstrate that the proposed method may maintain classification performance that is competitive with the original network while removing needless network parameter tuning in each training cycle. The trained (inference) model can use the dropout status of units as well.

They present an adaptive approach for training deep multilayer perceptron (MLP) networks that is superior to random dropout.

Because neural networks are computationally and memory heavy[4], they are challenging to implement on embedded devices. Furthermore, traditional networks set the design before training begins, preventing training from improving the architecture.

To overcome these restrictions, they[4] offer a strategy for learning only the necessary connections, which reduces the amount of storage and computing required by neural networks by an order of magnitude without reducing their accuracy. A three-step strategy is used to remove superfluous connections in this method.

Train the network to recognise important connections.

Then, prune the connections that aren't necessary.

Finally, retrain the network to fine-tune the remaining connections' weights.

The network is then retrained to learn the final weights for the remaining sparse connections in the final stage. This is a crucial stage. When the pruned network is utilised without being retrained, accuracy suffers dramatically.

We will discuss this point later in the results section itself which will include our own implementation of pruning some low magnitude weights in the fully connected layers of VGG-16 Model.

Paper[8] suggests that the model can be compressed once it has been initialized. Convolutional layers are notable for learning the key characteristics of the training dataset. Pruning the convolutional layer's

connections will reduce the model's accuracy. Furthermore, when compared to the connections of the convolutional layer, the fully connected layer's connections make up the majority of the model. As a result, we can prune only the completely linked layers' connections or eliminate connections of certain convoluted layers.

2.2.2 Conclusions Drawn from these papers

Pruning could lead to significant gain in computation as well as lead to efficient memory utilization.

As mentioned in these papers, pruning must be done carefully and in an iterative manner so as to not destroy the complete model itself.

Even in pruning, the model needs to be carefully retrained/ fine-tuned so as to achieve desirable accuracy

Since we are interested in compressing large models such as VGG-16 , AlexNet , ResNet for our project, we do not proceed with retraining/fine-tuning the network as the computation cost beared for the same is not justifiable here.

Hence, we do another literature survey on the paper “Siddharth Gupta et al. “ALigN: A highly accurate adaptive layerwise log2lead quantization of pre-trained neural networks”. In: IEEE Access 8 (2020),pp. 118899–118911[3]”.

3. Recent Work:

Since training of a large model such as VGG-16, AlexNet, ResNet requires a significant amount of time, we move forward with the idea of approximating a pre-trained model itself. We did a survey research on the thesis “ALigN: A highly accurate adaptive layerwise log2lead quantization of pre-trained neural networks. In: IEEE Access 8 (2020),pp. 118899–118911[3]” done by one of our senior Siddharth Gupta.

Below is brief summary of his thesis :

3.1 Motivation Behind Align and L2L

DNNs can withstand small errors. Some reduction in the precision of parameters in DNN has no discernible effect on network accuracy. Taking advantage of this aspect, recent research has demonstrated that employing various network approximation approaches, large-scale deep neural networks may be constructed with low computing and storage requirements.

However, in order to reduce the inaccuracy caused by approximation, these approximation strategies need computationally expensive network retraining. As a result, there is a need to design approximate procedures that do not require retraining.

They present a quantization-based approach for pre-trained deep neural networks to avoid the retraining and iteration process. Some researchers have developed network compression methods based on layerwise analysis. They also offer the parameter representation approach, which examines each DNN layer and further lowers parameter quantization error. Multiplication is the most computationally intensive operation in deep neural networks. Few studies make use of “bit-shift and addition” operations instead of multiplication in DNNs. For efficient DNN inference, this suggested approach additionally employs bit-shift and addition.

3.2 Log2Lead Quantization Technique

The suggested quantization approach L2L (Log2Lead) for DNN is described in this section. By selecting and storing the most significant 1's in the parameters of a pre-trained DNN, the proposed fixed-point quantization approach aims to reduce quantization-induced errors.

3.2.1 Proposed Template for Quantization

Recognizing the 1s that are more essential lowers quantization errors since the learned parameters are stored in 32-bit single-precision floating-point format. Histograms of the leading 1's in all the weights and biases of two distinct layers of the VGG-16 network are displayed in Fig2 to find critical 1's in Float32-based parameters. The leading 1 appears at various bit locations for most trained parameters (weights and biases), as illustrated; for example, the weights of Conv1 2 and Conv4 1 layers usually appear at bit positions -6 and -8, respectively. However, with the leading 1 at bit position -15, many weights have extremely low values.

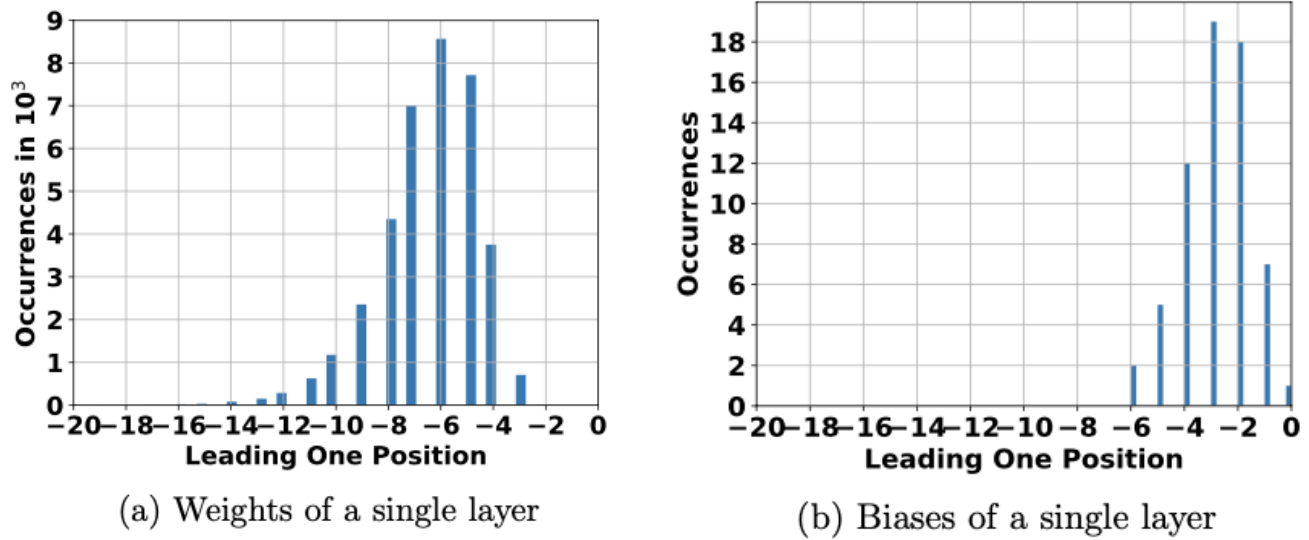


Fig2. "leading one" position of weights and biases in Conv1 layer[Ref. from [3]]

They use log2 to detect the position of the leading 1 in a fraction in the suggested log2lead approach. However, log2lead observes the following bits placed after the leading 1 position to reduce quantization problems. Fig3 depicts the template for the suggested log2lead quantization approach for N-bit quantization. The first bit is used to display the quantized parameter's sign. Following $\lceil (N-1)/2 \rceil$, two bits are set aside to store the position of the leading 1 in the non-quantized parameter. After the leading 1, the remaining bits are utilised to hold the values of the following $\lfloor (N-1)/2 \rfloor$ bits .

The $\lfloor (N-1)/2 \rfloor + 1$ bit rounding also aids in maintaining the accuracy of a rounded 2 number. Fig4 demonstrates how to use the suggested log2lead approach to quantize a fractional value of 0.217884. The leading 1 is located at bit position 3, which is recorded as the binary integer 0011 in our template. The next 4 bits are evaluated and are then rounded off to binary pattern 110 after the leading 1. The quantization value for this is 0.21875.

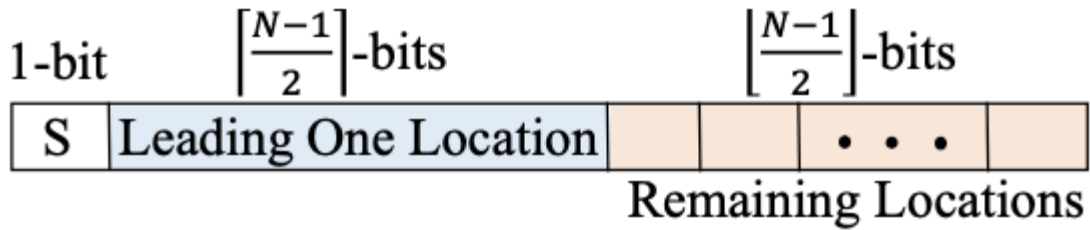


Fig3. Proposed Template for Log2Lead Quantisation[Ref. from [3]]

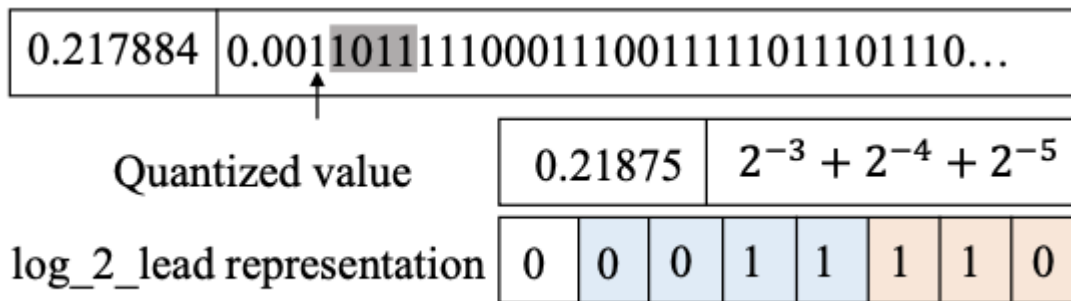


Fig4. Example of Log2Lead Quantisation[Ref. from [3]]

3.3 ALigN: Layerwise Quantization of Parameters of Pre-trained Neural Networks

The adaptive layerwise modification of L2L approach, ALigN, is described in this section. The ALigN approach provides various templates for DNN parameter representation, substantially reducing quantization-induced error.

3.3.1 Proposed Template for Quantization

Quantization-induced errors can be reduced by properly utilising the available quantization bit-width. Despite the fact that the suggested L2L approach reduces quantization-induced errors, it uses a fixed template to represent the quantized parameters of all layers in a pre-trained DNN.

The author observed the distribution of parameters of the various layers in the pre-trained DNN. Post observation, it was evident that the fixed $\lceil (N-1)/2 \rceil$ bits for storing the leading 1 positions were not utilised efficiently in some layers.

The “leading 1” histogram for the weights and biases of the Conv1_1 layer of pre-trained VGG-16 Model, for example, is shown in Fig2. The leading 1 for these characteristics appears to be present most frequently at “bit position 3”. When $N = 8$ bits is used, bit position 3 may be recorded in two bits (the magnitude of the bit position is stored), whereas log2lead utilizes a predetermined template of four bits to store leading 1 position.

'ALigN,' an adaptive log2lead quantization approach can overcome this problem. As shown in Fig5, the ALigN approach may provide numerous combinations of "leading one" location storage and following bits storage. The layerwise analysis for selecting acceptable ALigN setups is also described.

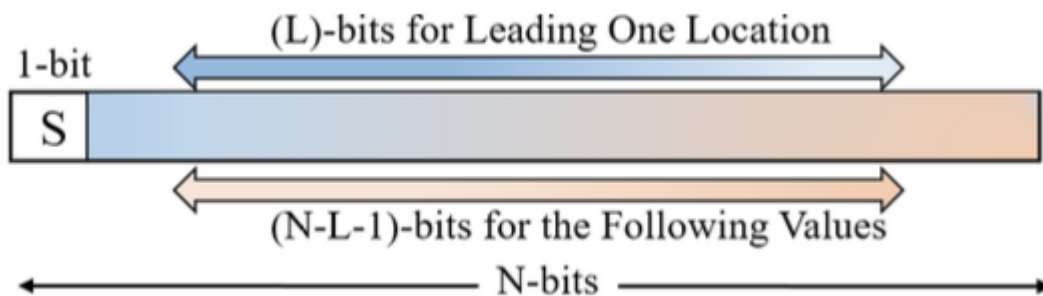


Fig5. Proposed Template for AligN Quantisation[Ref. from [3]]

When we Align the “Log2Lead” scheme inorder to fit the most appropriate representation for the parameters of each layer in a pre-trained neural network, we see that the induced error after the quantisation seems to reduce drastically.

To determine the ideal amount of bits L_w and L_b for the purpose of storing the layer parameters, we make use of the below mentioned equations. Post this step, we then use the remaining “N-L-1” bits to store the remaining ones of the respective parameter.

The modified template of the suggested quantization system, ALigN, to represent the float32 bit weights in “N bit” template is shown in Fig5

4. Proposed Methodology:

In the Align algorithm, the author has quantized all the weights of pre-trained neural networks.

In post training quantisation, there is a significant drop in accuracy of a model. This is due to the fact that we go from float-32 bit representation to some lower bit representation [8 bit in Align].

This error induced after the quantisation could lead to significant challenges in the inference stage

To overcome this, we decided to quantise only some of the weights using sampling techniques

To do this, we use Sampling Algorithm to select some sample of weights from the entire distribution and then quantize them using the ALigN technique.

The Sampling technique we considered for doing this is Linear Systematic Sampling.

4.1 Linear Systematic Sampling

In “systematic sampling”, everyone in the target population has an equal chance of being picked. Each of the 100 people in the population has a one-in-a-hundred probability of getting selected for the sample group.

“Systematic sampling” is the process of selecting a sample group from a target population based on a predetermined interval.

The two key phases in creating systematic sampling are as follows:

1. We divide the size of the target population "N" by the sample size "n" to get the sampling interval. If the value is in decimals, it must be rounded to the nearest whole number or integer.
2. Beginning with a random starting point "r," the sample interval I is then used to choose respondents from the target population. To prevent picking a biased sample group, researchers must check that the sample frame's list is not ordered in a cyclical or periodic manner before selecting the sample group.

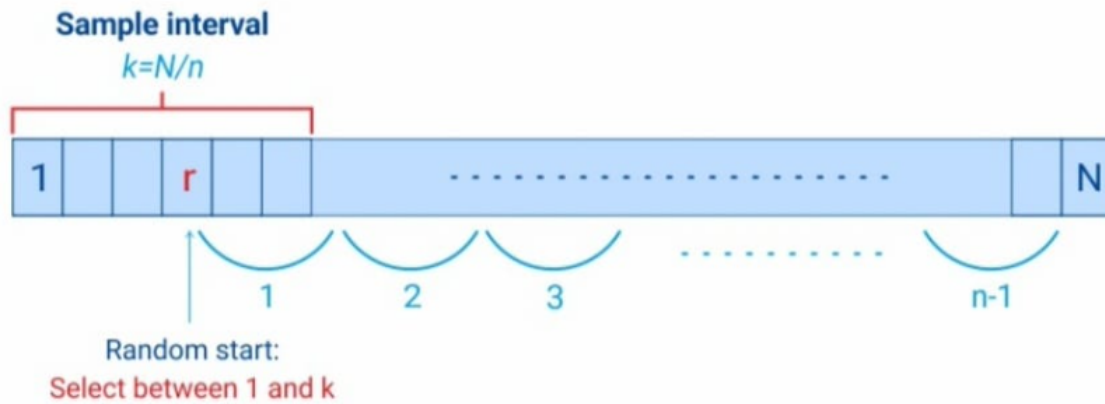


Fig6. Example of Linear Systematic Sampling

4.2 Advantages of using Linear Systematic Sampling

- (i) When the sample frame's list is arranged randomly, the possibility of bias is minimised.
- (ii) Complexity of linear time $O(N)$
- (iii) Other probability sampling methods, such as cluster sampling and stratified sampling, or non-probability approaches, such as convenience sampling, run the danger of creating highly biased clusters, which systematic sampling avoids since the members are at a constant distance from one another.
- (iv) Data without a pattern: There are some pieces of information that aren't organised. Using systematic sampling, this data can be evaluated in an unbiased manner.
- (v) In research, there is a low probability of data manipulation: It is extremely useful when researching a vast topic, especially when the risk of data manipulation is minimal.

4.3 Pseudocode for Linear Systematic Sampling

Algorithm 1 Linear Systematic Sampling

Input: Model, Skipsize

Output : Quantized Model

Sample Size $\leftarrow N / \text{Skipsize}$

StartIndex $\leftarrow \text{rand}(1, \text{Skipsize})$

Counter $\leftarrow 1$

while $i \neq N$ **do**

if *Counter* = *StartIndex* **then**

Weight $\leftarrow \text{QuantizedWeight}$

Counter $\leftarrow \text{Counter} + 1$

if *Counter* > *Skipsize* **then**

Counter $\leftarrow 1$

$i \leftarrow i + 1$

Model $\leftarrow \text{Quantized Model}$

return *Model*

Fig7. Pseudo Code for Linear Systematic Sampling

4.4 Flowchart explaining the working of Linear Systematic Sampling

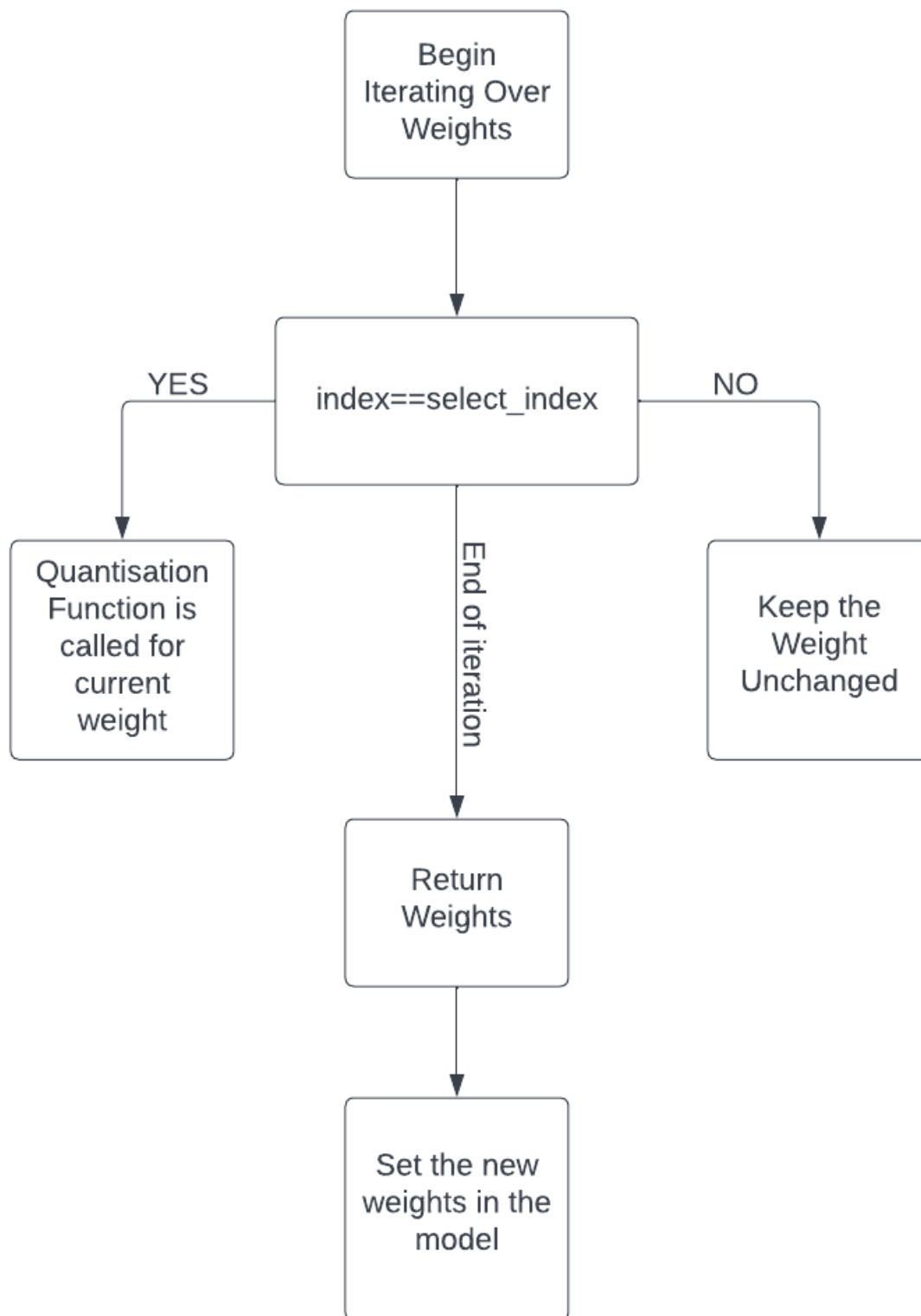


Fig8. Flowchart for Linear Systematic Sampling

5. Results and Experimentation:

5.1 Sampling And Quantisation

Model Considered : VGG-16

Dataset : Imagenet Validation Dataset 2012

For experimentation we have written our own code for quantising the weights of VGG-16 model using ALigN technique. For AligN quantisation , we use $N=8$ bits for storing the weights of pretrained VGG-16 model in the AligN template. We iterate over the weights of VGG-16 model one by one and if the current index of the weight matches that with the select index , we quantise that particular weight value using AligN quantisation.

For AligN quantisation, we find the suitable l_w and l_b bits to efficiently store the ““leading one”” value of the weights and biases by iterating over l_w and l_b values [1 to 7] . We find the best l_w and l_b bits by calculating the minimum average error for the corresponding weight and biases value.

The best suitable l_w and l_b bits are then used to store the ““leading one”” values of the corresponding weights. This is done for each layer iteration so that we can optimally store the "leading one" bit position using the minimum no of bits.

The remaining $N-L-1$ bits are then used for storing the remaining ones for the same.

Sampling (%)	Top-1 Accuracy	Top-5 Accuracy
25	64.86	85.72
50	64.80	85.62
75	64.62	85.56
100	64.58	85.58

Table1. Result for Linear Systematic Sampling applied over VGG-16 Model

Image ID	Actual Labels	25% Sampling Pred	100% Sampling Pred
 ILSVRC2012_val_00000263	Saluki, gazelle hound	Saluki, gazelle hound	Great Dane
 ILSVRC2012_val_00000778	suspension bridge	suspension bridge	picket fence, paling
 ILSVRC2012_val_00000925	green lizard, Lacerta viridis	green lizard, Lacerta viridis	grasshopper, hopper
 ILSVRC2012_val_00000351	window screen	window screen	window shade
 ILSVRC2012_val_00000254	mobile home	mobile home	freight car

Fig9. Top 1 Mismatch recorded for 25% Sampling Vs 100% Sampling



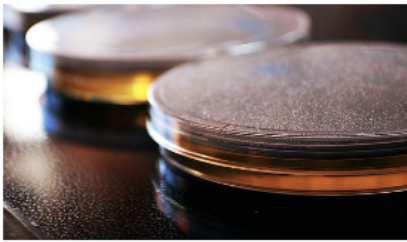

Image ID	Actual Labels	25% Sampling	100% Sampling
 ILSVRC2012_val_00000365	moving van	['garbage truck', 'crane ', 'Harvester', 'trailer truck', ' <u>moving van</u> ']	['garbage truck', 'crane', 'Harvester', 'trailer truck', 'tow truck']
 ILSVRC2012_val_00000419	envelope	['hard disc', 'binder', 'laptop', ' <u>envelope</u> ', 'printer']	['laptop', 'hard disc', 'inder', 'printer', 'notebook']
 ILSVRC2012_val_00000465	Petri dish	[' <u>Petri dish</u> ', 'espresso', 'Loafer', 'beer glass', 'pill bottle']	['pill bottle', 'bottlecap ', 'espresso ', 'beer glass', 'beer bottle ']
 ILSVRC2012_val_00000854	wig	['pedestal', ' <u>wig</u> ', 'hourglass', 'table lamp ', 'vase ']	['pedestal', 'table lamp', 'Brabancon griffon', 'stove', 'throne']

Fig10. Top 5 Mismatch recorded for 25% Sampling Vs 100% Sampling

	Layer	Conv 1_1	Conv 1_2	Conv 2_1	Conv 2_2	Conv 3_1	Conv 3_2	Conv 3_3	Conv 4_1
Bits for storing leading one	25 % Sampling	4	4	5	5	5	5	5	5
	50 % Sampling	4	5	5	5	5	5	5	5
	75 % Sampling	4	5	5	5	5	5	5	5
	100 % Sampling	4	5	5	5	5	5	5	5

	Layer	Conv 4_2	Conv 4_3	Conv 5_1	Conv 5_2	Conv 5_3	FC6	FC7	FC8
Bits for storing leading one	25 % Sampling	5	5	5	5	5	5	5	5
	50 % Sampling	5	5	5	5	5	5	5	5
	75 % Sampling	5	5	5	5	5	5	5	5
	100 % Sampling	5	5	5	5	5	5	5	5

Table2. Optimal no of bits selected by AlignN Quantisation for storing leading1 parameters VGG-16

5.2 Pruning to remove some of the redundant weights in FC layers of VGG-16

As discussed in Section 2.2.1 , we have also tried to implement the concept of pruning the redundant weights of VGG-16 mostly present in the dense fully connected layers.

Due to the time constraint, we have not extended our approach to prune in an iterative manner but we have tried to keep the threshold value to as low as 0.0001.

But as the papers suggest, pruning without retraining leads to significant accuracy drop in the model itself. This was evident from our observations tabulated below.

However, we have a strong intuitive idea that pruning along with retraining/fine-tuning could be a great idea to proceed with. This could also be coupled with using quantisation on the model post pruning. Quantising the pruned model could lead to significant model compression

Pruning Threshold	Top-1 Acc	Top-5 Acc
1	0.06	0.45
0.0001	0.1	0.48

Table3. Results of pruning in the FC layers of VGG-16 Model

6. Conclusion and Future Work:

Through the course of this project, we have implemented :

- (i) “ALigN / L2L” Quantisation technique
- (ii) Linear Systematic Sampling
- (iii) Unstructured Pruning

The results we obtained shows that models such as VGG-16 and ResNet are robust to 8 Bit quantisation. These models show little to no improvement when sampling techniques such as Linear Systematic Sampling are applied to it.

Therefore the tradeoff between the gain in accuracy and cost of storing the 32 bit floating weight is not justified in this case.

Through our research and experimentation, we found that some models such as MobileNet v1 are extremely sensitive to quantisation. Advanced Sampling Techniques such as Pivotal and Cube Sampling can be applied and tested on this model to see if there is a significant gain in the accuracy of the model post quantisation.

We have also researched a bit about pruning. Unstructured Pruning, where we do not completely drop the weight of a CNN model but replace it with zero can be tried in an iterative manner. These zeroed weights can later be completely dropped off from the model, leading to significant model compression.

Quantisation post pruning can also be an effective technique for compressing large CNN models.

7.References:

- [1] Angela Fan et al. “Training with quantization noise for extreme model compression”. In: arXiv preprint arXiv:2004.07320 (2020).
- [2] Amir Gholami et al. “A survey of quantization methods for efficient neural network inference”. In: arXiv preprint arXiv:2103.13630 (2021).
- [3] Siddharth Gupta et al. “ALigN: A highly accurate adaptive layerwise log2lead quantization of pre-trained neural networks”. In: IEEE Access 8 (2020), pp. 118899–118911.
- [4] Song Han et al. “Learning both weights and connections for efficient neural network”. In: Advances in neural information processing systems 28 (2015).
- [5] Benoit Jacob et al. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 2704–2713.
- [6] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. “Survey of dropout methods for deep neural networks”. In: arXiv preprint arXiv:1904.13310 (2019).
- [7] Muhammad Bintang Gemintang Sulaiman, Kai-Cheung Juang, and Chih-Cheng Lu. “Weight Quantization in Spiking Neural Network for Hardware Implementation”. In: 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan). IEEE. 2020, pp. 1–2.
- [8] Yushuang Yan and Qingqi Pei. “A robust deep-neural-network-based compressed model for mobile device assisted by edge server”. In: IEEE Access 7 (2019), pp. 179104–179117.
- [9] Stone Yun and Alexander Wong. “Do All MobileNets Quantize Poorly? Gaining Insights into the Effect of Quantization on Depthwise Separable Convolutional Networks Through the Eyes of Multi-scale Distributional Dynamics”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 2447–2456.

- [10] Rongzhao Zhang and Albert CS Chung. "MedQ: Lossless ultra-low-bit neural network quantization for medical image segmentation". In: Medical Image Analysis 73 (2021), p. 102200.
- [11] Timur R Zhangirov et al. "Forward Propagation Neural Network Weighting Analysis as a Model Estimation Method". In: 2021 II International Conference on Neural Networks and Neurotechnologies (NeuroNT). IEEE. 2021, pp. 10–12
- [12] <https://www.questionpro.com/blog/systematic-sampling/>
- [13] Vanhoucke, Vincent, Andrew Senior, and Mark Z. Mao. "Improving the speed of neural networks on CPUs." (2011).
- [14] Lee, Anzy, Zong Woo Geem, and Kyung-Duck Suh. "Determination of optimal initial weights of an artificial neural network by using the harmony search algorithm: application to breakwater armor stones." Applied Sciences 6.6 (2016): 164.
- [15] Ajao, Oluwaseun, Deepayan Bhowmik, and Shahrzad Zargari. "Fake news identification on twitter with hybrid cnn and rnn models." Proceedings of the 9th international conference on social media and society. 2018.
- [16] Salehinejad, Hojjat, and Shahrokh Valaee. "Ising-dropout: A regularization method for training and compression of deep neural networks." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.