ASSIGNMENT – 3

SYSTEMS ENGINEERING FOR DEEP LEARNING

- **Shreyas S (EP22B040)**

Q1)

A) Prepare CIFAR-10 with proper normalization and data augmentation; specify transforms. (5)

- As shown in the *train_baseline.py* file, the CIFAR-10 dataset is prepared with sufficient data augmentation to enable the training of the mobilenetv2 model.
- The various transforms performed are:
  - Random-crop
  - Horizontal-flip
  - Normalization
  - Label Smoothing (ε=0.1)
  - Mix-up augmentation (α=0.2)
  - Weight decay (5e-4): L2 regularization of weights

B) Describe your MobileNet-v2 configuration (e.g., width multiplier, dropout, BN settings) and training strategy (optimizer, LR schedule, regularization, epochs, batch size)

- Reduced stride from 2 → 1 in early layers (conv1 and first residual block with expansion=6, out_planes=24). This preserves higher spatial resolution for CIFAR-10's small 32×32 images.
  - CIFAR-10 is a relatively simpler dataset to train on, compared to the original imagenet, hence higher spatial resolution captures more features to train.
- No width multiplier / Dropouts have been used
- Batch Normalization – applied after every convolutional layer (at both depth wise and pointwise granularity)
- Residual connections: Retained wherever input/output channels match and stride=1.

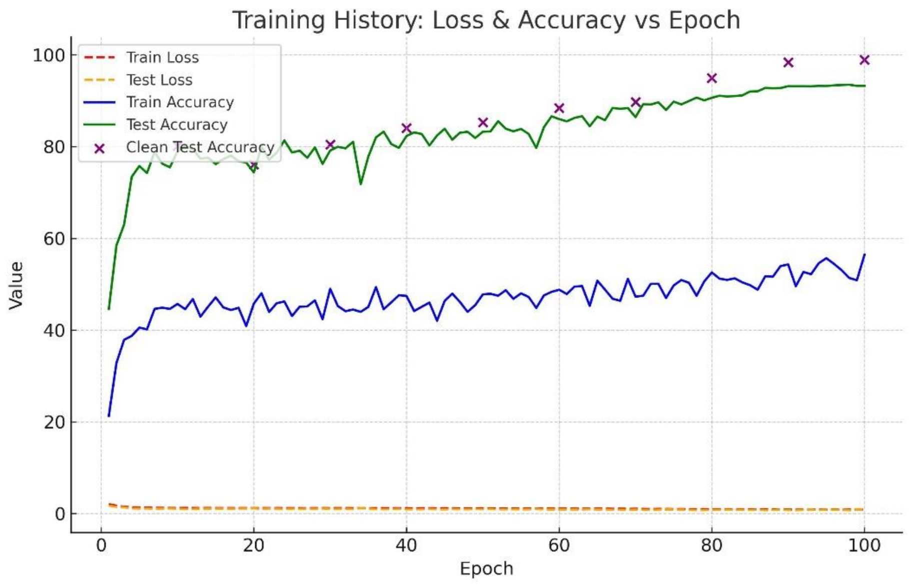
Training Strategy:

- Optimizer: Stochastic Gradient Descent
  - Momentum=0.9
  - Nesterov = True

- Learning rate Schedule:
    - Warmup for 5 epochs (linear increase from 0 to base LR)
    - Cosine annealing decay afterwards (applied once per batch step)
- Batch size used: 128
- Total Epochs trained for (to hit baseline) : 100

C) Report final test top-1 accuracy and include loss/accuracy curves.

- **Baseline Accuracy** achieved after 100 epochs (accuracy saturated for the last few epochs): **93.26%**



- The train accuracy in blue is an artifact of the heavy data augmentation/regularization that is being applied on the training dataset.
- The clean test accuracy (plotted only at the saved checkpoints), refers to forward pass, ie: inference accuracy over un-augmented CIFAR10 dataset.

Q2) Model Compression Implementation

A) Implement a configurable compression method for reducing both model weights and activations. Clearly explain your design choices.

- A 2-stage compression pipeline approach is taken where:
    - Step 1: **Prune the network** to achieve a decent sparsity for negligible drop in accuracy (~1%)
    - Step 2: Perform **quantization**

Pruning the network:

- Inspired by AutoSlim (work on pruning – shown to work reasonably well for mobilenetv2)
    - Granularity: channel removal
- The core criteria used for deciding the sensitivity of a channel is by setting a threshold on the scaling factor (γ) in the batch normalization layers. This gives a fair description of the entire channel – which turns out to be better than just L1 norm.
- The model is then fine-tuned by retraining for a few epochs (~10).
- The paper also suggests to modify the Loss function for fine-tuning:
    - Adding L1 regularization to the cross-entropy loss function, seems to produce a better convergence
- The whole process of pruning is done iteratively, until a target sparsity is achieved:
    - Prune some channels based on sensitivity
    - Finetune the remaining network.
    - Perform the finetuning by setting masks to the pruned channels (they cannot become non-zero during gradient descent)
- A new smaller model is eventually created – without the pruned channels.

Quantization of the Network:

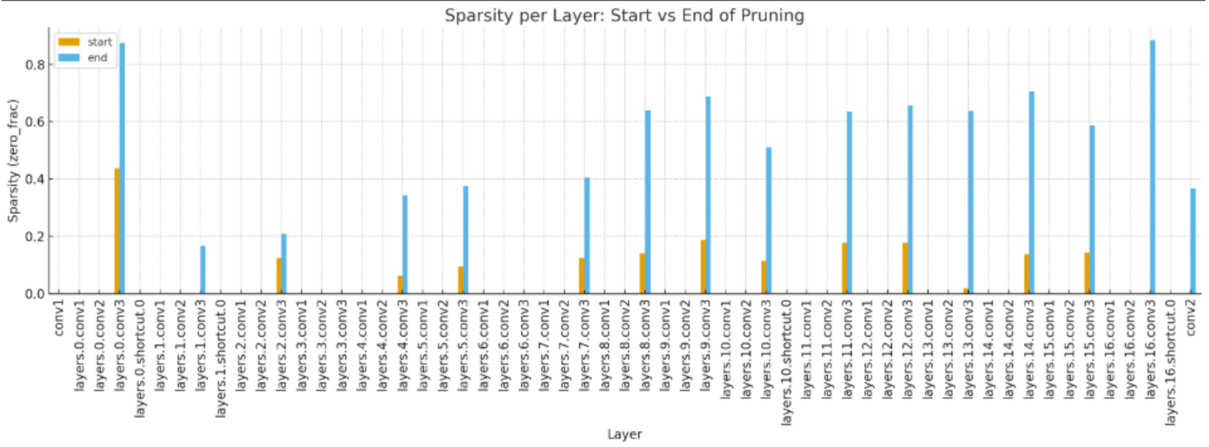
- K-Means cluster is implemented on the pruned & retrained network. Hence, Post Training Quantization is the approach taken for quantizing the network.
- The K-Means Cluster is implemented just as described in the *Deep Compression* paper.
- At a layer-wise granularity, centroids are assigned for the bins. The number of centroids is fixed for all convolutional layers and all Fully Connected layers (each of them can differ though).
- As per the paper's conclusion, the centroids are initialized in a symmetric linear fashion to achieve the highest compression for fixed drop in accuracy.
- The centroids are fine-tuned iteratively: the WCSS is minimized as the objective function. To perform this, gradient descent is applied and the quantization is estimated by Straight-Through Estimation (setting grad = 1 for quantization units)
- Final compressed model is stored using a code-book.

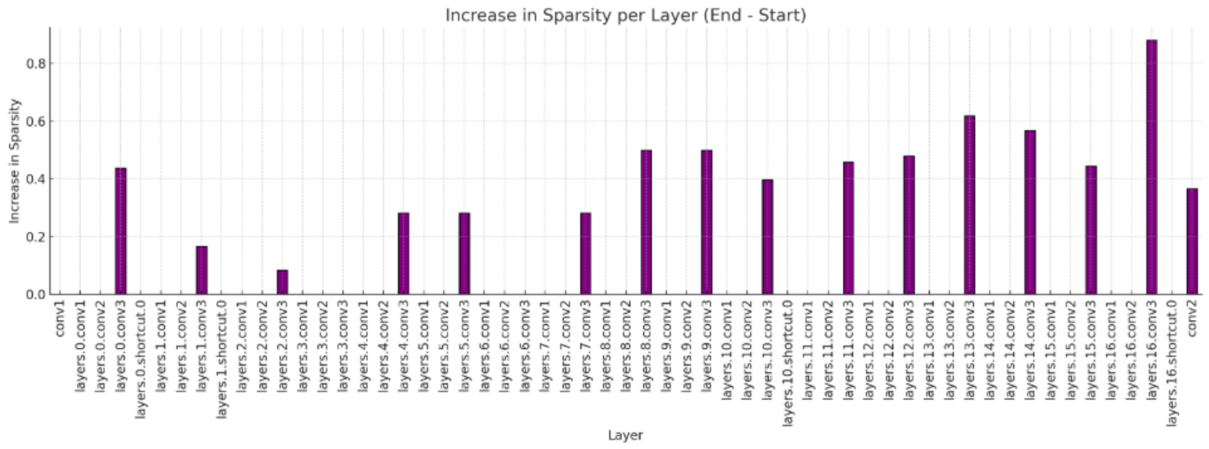
Input Activations:

- Possibly due to the high quantisation noise introduced by the pruning layers, the quantization of input activations even to int8 suffers with significant loss in accuracy.
- Hence, the input activations are not explicitly compressed for the current execution!

B) Show how the compression is applied to MobileNet-v2 (which layers compressed, any exceptions).

Layer-Specific Compression is applied only at the pruning stage.





For ~ 1% drop in accuracy, a channel count wise sparsity of 50% and global sparsity of ~36% can be achieved. (The accuracy of the resulting smaller network is 92.11%)

Post-Pruning Quantization:

The compression ratio after pruning is calculated as per the formula proposed in the paper: **$n*b / (\log_2(k) + k*b)$**

Accordingly, it is observed that the FC layer is tolerant to an aggressive compression, while the CONV layers are also amenable to a significant quantization without much loss in accuracy.

Accordingly, for different levels of quantization:

FC – Quantization	CONV Quantization	Compression(total)	Accuracy (%)
4	4	19.48	91.54
2	4	19.57	91.04
2	2	39.07	87.48
1	2	39.23	87.38

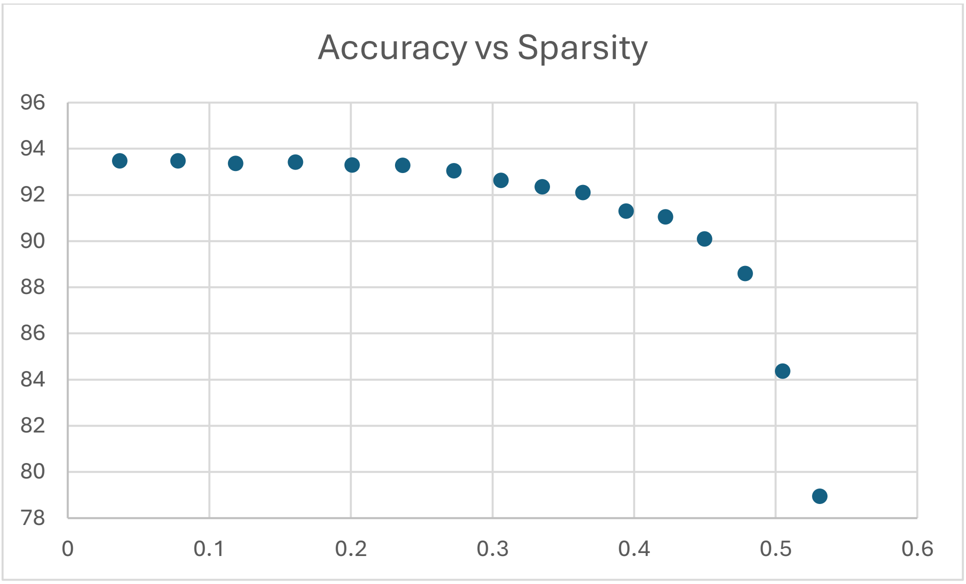
The compression (total) in the above table is overall compression of weights after both pruning and quantization. This is calculated by the product of inverse sparsity achieved and compression from quantization.

C) Document storage overheads (e.g., metadata, scaling factors) and include them in size estimates.

- Over head from compressed storage of pruning – either bitmap encoding or CSR/CSC representation. Otherwise the overall size of the .pth model file remains the same (as observed).
- For quantization, the centroids for each layer becomes a significant contributor to meta-data.
- If successful input activations are used, the scale & zero point for each layer will also correspond to meta-data to be accounted for.

Q3) Compression Results

A) Accuracy for various global sparsity can be drawn as:

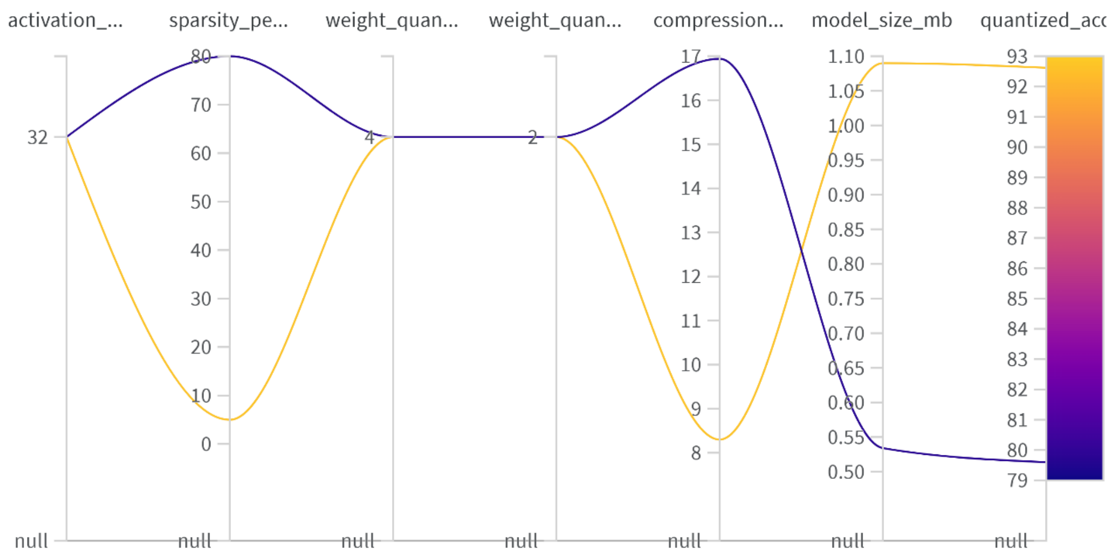


Since ~1% is is the drop until about 3.6%, that is the sparsity chosen.

To explicitly witness reduction in model size, a bitmap or CSR encoding is performed which will increase meta-data but decrease the overall size of the model.

B) WandB Plot

WandB was generated/iterated over all the cases. However, the final plot was logged only for the 2 extreme cases. As time is expensive, I was unable to perform the entire loop again to capture the full parallel coordinate plot. The following is the captured plot:



Note: the compression mentioned in this plot is wrt a pruned model – not an unpruned model.

Question 4)

Establishing the Baseline:

Total Pruneable Channels	2,784	Calculated in the pruning script.
Total Model Weights (N_{total})	~2,246,000	Approx. parameters in the custom MobileNetV2 architecture.
Storage per Weight	3 bits (4 Bytes)	Standard FP32 format.
Original Model Size (MB)	~8.57 MB	This is the true floating-point baseline size.

As established in Question 2, the compression achieved depends on the desired accuracy. If the **application is very sensitive**:

Compression of weights = 19.48 for 91.54% accuracy.

However, if **aggressive compression** can be applied for less sensitive application:

Compression of weights = 39.23 for 87.38% Accuracy

- Using bitmap encoding, each weight would be mapped to one extra bit for compressed representation (after pruning)
- Since each weight takes 4 bits in the sensitive case, divide the compression by a factor of 1.25

Therefore,

Compression of model ~ 19.48/1.25 = 15.58

The expected model size therefore would be = 8.57 MB / 15.58 ~ 0.55MB!!

Link to GitHub Repo for source code:

https://github.com/Shreyaassss/CS6886-Asgn3.git