Advanced Image Processing: Assignment 4

JPEG Implementation :

1.

Algorithm :

- Computed the DCT transform of greyscale image cameraman.tif by applying 8x8 discrete cosine transform (DCT) for every non-overlapping block.
- Quantized each DCT coefficient in an 8x8 block with the quantisation table given in the problem.

$$y(i,j) = \left\lfloor \frac{x(i,j)}{Q(i,j)} + 0.5 \right\rfloor$$

- Encoded each quantized index appropriately, as mentioend below

Quantized DCT index	Code
0	0
-1,1	10x
-3,-2,2,3	110xx
-7,-6,-5,-4,4,5,6,7	1110xxx
...	...

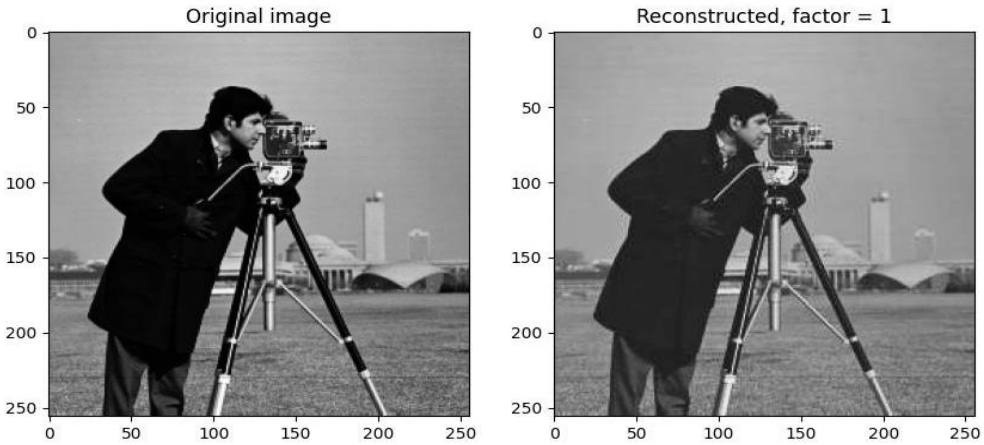
- The reconstructed image is obtained by taking the inverse DCT for each block of quantized reconstructions of DCT coefficients.

Note :

- Encoding of DCT coefficients is a lossless method, i.e. decoding the information does incur any loss of data.
- But reconstruction of the DCT coefficients from quantised values incurs a loss of data

$$\hat{x}(i,j) = y(i,j)Q(i,j).$$

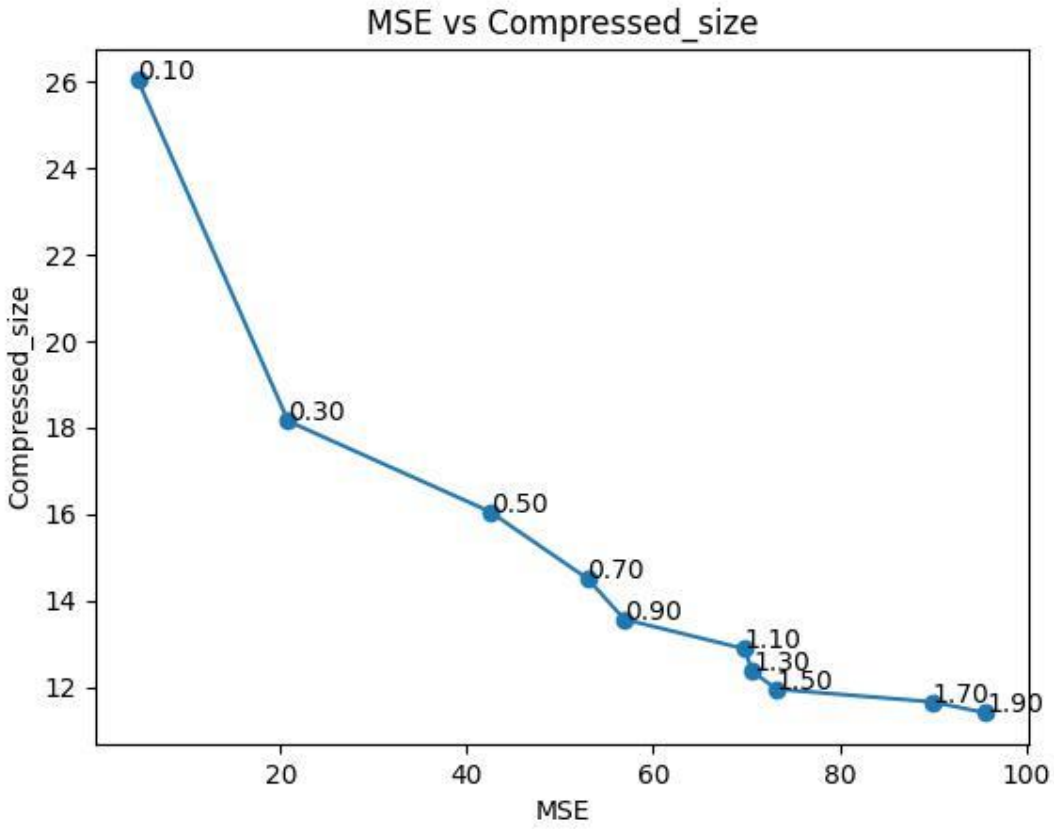
Results :



- MSE between the original and reconstructed image for the case of the given Quantization table is: *54.209869384765625*
- Size of the compressed file is in Kilobyte(KB): *13.21484375*
- Compression ratio is(defined as the ratio of the input image in bits and size of the output file in bits) : *4.8430387230269*
- We can infer that the reconstruction of DCT coefficients being lossy has led to some non-zero MSE between the original and reconstructed image.
- Through JPEG compression, we also decreased our image size by four times(approximately) but at the cost of losing information(the coat of the reconstructed image got faded compared to the original).

2.

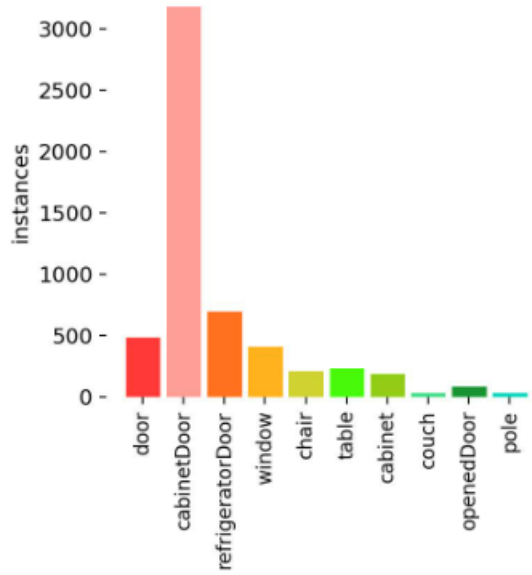
The plot of MSE(between original and reconstructed) and Compressed file size:



- The annotated value at each point represents the factor with which the given quantisation table is multiplied.
- If the elements of the quantisation matrix are smaller, then
    - The quantised indices become large, which in turn increases the compressed file size.
    - Smaller range of DCT coefficients quantised to the same quantised index value, that implies reconstruction of DCT coefficients from quantized indices doesn't incur a significant loss.
- If the elements of the quantisation matrix are larger, then
    - The quantised indices become smaller, which in turn, decreases the compressed file size.
    - Larger range of DCT coefficients quantised to the same quantised index value implies that reconstruction of DCT coefficients from quantised indices incurs significant loss.
- So, as we decrease the quantisation matrix values, we end up getting larger MSE(between original and reconstructed) and smaller compressed file size.

YOLO Object Detection :

Statistics of the number of objects per each class in the given dataset :



- Pole and couch have less number of objects in the dataset, typically 1-100.

Training :

- Used the pre-trained Yolov5 model implementation available at https://github.com/ultralytics/yolov5.
- Initialized the model with pretrained weights **yolov5m.pt** and chose the medium architecture of Yolov5 i.e. yolov5m.yaml.
- All the images are resized to 640*640 size(Done automatically by the utils present in Yolov5).
- Trained the model for 200 epochs with batch size 128, using a distributed GPU setting with distributing 128 examples among 4 GPUs.
- Command run for training the model:
    - `git clone https://github.com/ultralytics/yolov5  # clone`
    - `cd yolov5`
    - `pip install -r requirements.txt  # install`
    - `python -m torch.distributed.run --nproc_per_node 4 train.py --img 640 --batch 128 --epochs 200 --data object_detection/data.yaml --weights yolov5m.pt --cfg models/yolov5m.yaml --name yolov5m --device 4,5,6,7`
- Command run for testing the model:
    - `python val.py --weights runs/train/yolov5m/weights/best.pt --task test --data object_detection/data.yaml --device cuda:6`

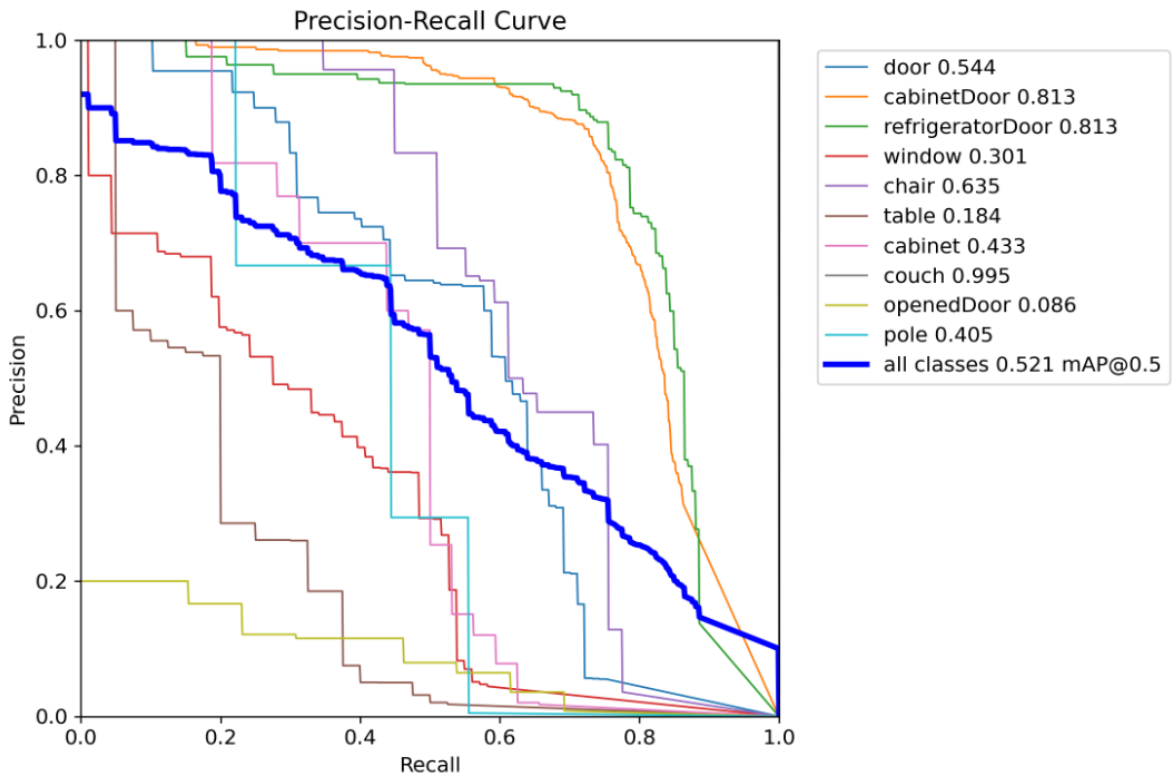
Results :

Ground truth :

Predicted :

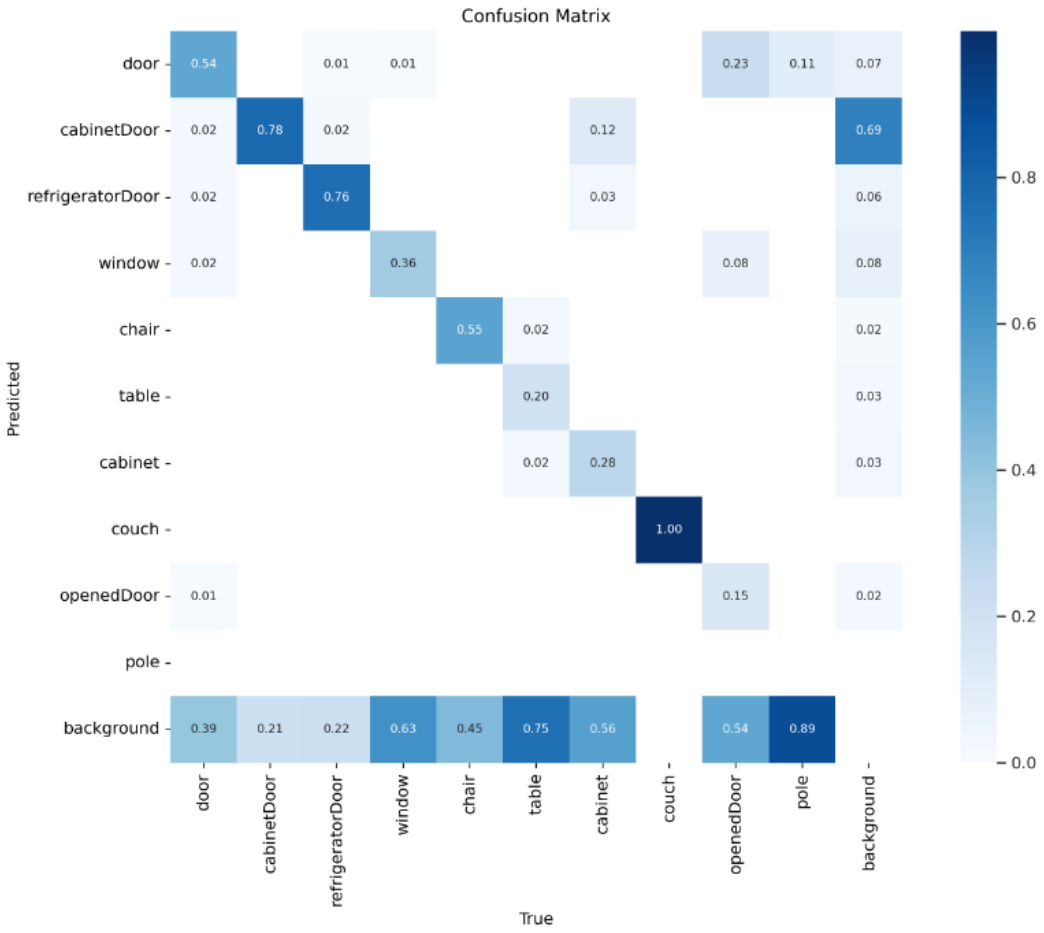
- The predictions made by our model are almost correct, but it missed out on some of the ground truth objects.

Precision-Recall curve :



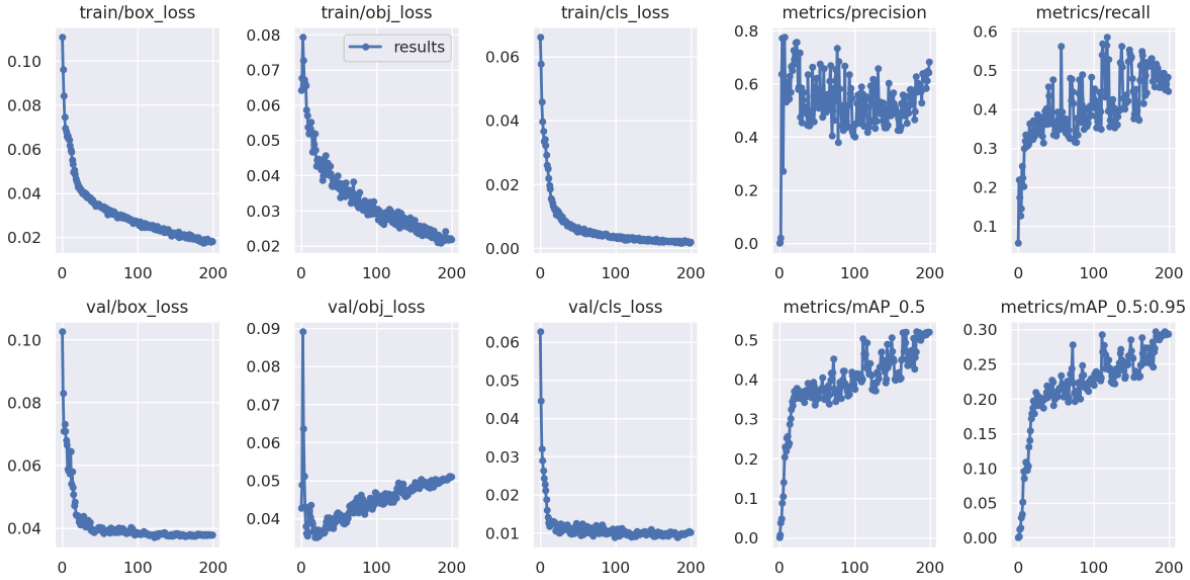
- The value to the right of each class in the table indicates the Area under the corresponding PR curve. The higher the Area under the PR curve, is better.
- As the IoU threshold decreases, recall increases and precision decreases.
- We can observe that the Area under the PR curve is higher for classes refrigerator and cabinetDoor, it follows from the fact that number of instances of both classes are relatively high.

Confusion matrix :



- Trained model is performing well on the classes of objects whose count is more in training data(i.e. cabinetDoor,refrigerator,door).
- Most of the predicted objects are correct, but our model did not detect a significant amount of objects (predicted as background means nothing is predicted).
- We can possibly address this problem by increasing the weight on objectness loss.

Plots of losses and mAP on train and val dataset across 200 epochs :



- We can observe that loss decreases as we go along each epoch.
- Increase in the validation objectness loss is evident that most objects are not detected in our model.
- After running for 200 epochs, achieved mAP[0.5] = 0.5 and mAP[0.5:0.95] = 0.32.
- There is no significant improvement observed by training the model for more epochs.

Validation set results :

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	230	1289	0.619	0.449	0.503	0.313
door	230	97	0.679	0.433	0.519	0.318
cabinetDoor	230	765	0.755	0.749	0.819	0.456
refrigeratorDoor	230	192	0.851	0.711	0.802	0.539
window	230	91	0.457	0.323	0.301	0.16
chair	230	49	0.825	0.385	0.558	0.328
table	230	40	0.358	0.2	0.129	0.0731
cabinet	230	32	0.578	0.344	0.356	0.233
couch	230	1	0.441	1	0.995	0.796
openedDoor	230	13	0.25	0.0769	0.118	0.0451
pole	230	9	1	0.269	0.43	0.183

Testing set results :

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	107	550	0.654	0.27	0.325	0.193
door	107	34	0.677	0.176	0.31	0.15
cabinetDoor	107	179	0.662	0.374	0.396	0.139
refrigeratorDoor	107	2	0.505	1	0.995	0.746
window	107	63	0.584	0.222	0.314	0.202
chair	107	87	0.738	0.291	0.352	0.214
table	107	47	0.487	0.17	0.224	0.116
cabinet	107	52	0.542	0.115	0.187	0.123
couch	107	58	0.625	0.172	0.267	0.107
openedDoor	107	20	0.817	0.05	0.0762	0.0394
pole	107	8	0.905	0.125	0.129	0.0907

- It is evident from the above results that classes having more instances in training data(cabinetDoor, refrigeratorDoor) have got better precision, recall and mAP values.