Running Deep Learning Model On RaspberryPi

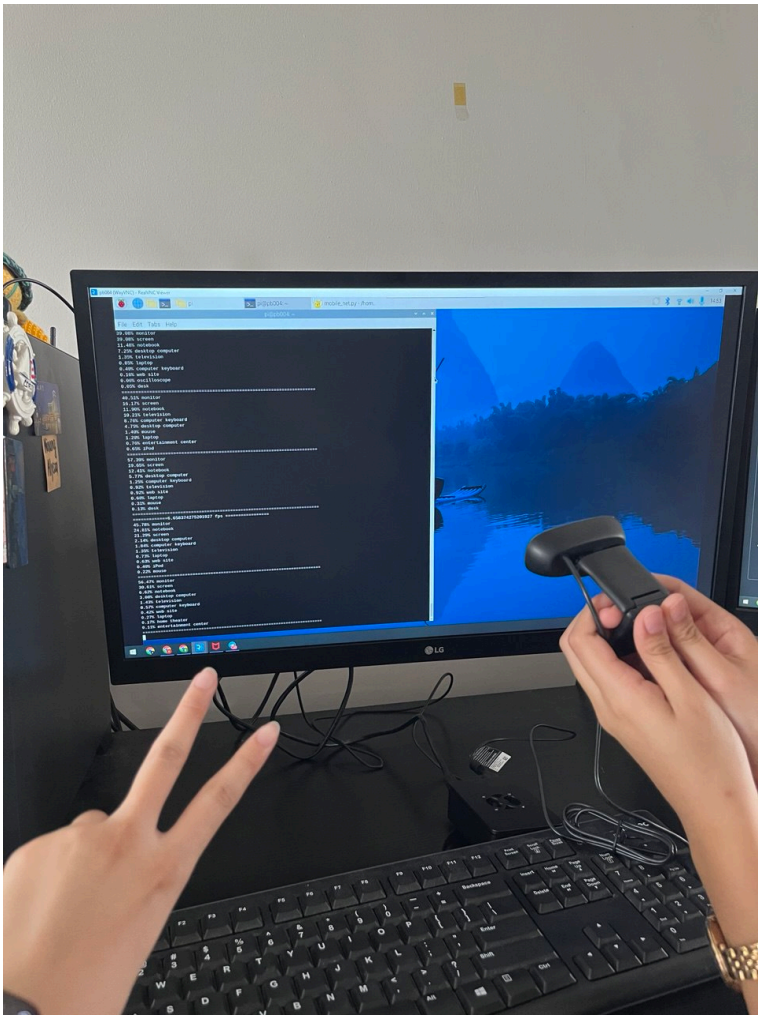
```
(dlonedge) pi@pb004:~ $ python3 mobile_net.py
/home/pi/dlonedge/lib/python3.11/site-packages/torchvision/models/_utils.py:208:
d' is deprecated since 0.13 and may be removed in the future, please use 'weights
  warnings.warn(
/home/pi/dlonedge/lib/python3.11/site-packages/torchvision/models/_utils.py:223:
eight enum or `None` for 'weights' are deprecated since 0.13 and may be removed 1
 equivalent to passing `weights=MobileNet_V2_Weights.IMAGENET1K_V1`. You can also
FAULT` to get the most up-to-date weights.
  warnings.warn(msg)
=============4.254323139994413 fps ================
=============5.598692042420078 fps ================
=============5.589069279980882 fps ================
=============5.564439695963575 fps ================
=============5.605612155571861 fps ================
=============5.598180167185795 fps ================
=============5.662895801853517 fps ================
=============5.585157040640591 fps ================
=============5.6044711369722195 fps ===============
=============5.568085174642754 fps ================
=============5.613910352590657 fps ================
=============2.912200127847938 fps ================
=============3.6970734945037123 fps ===============
=============3.439689449882297 fps ================
=============4.419182032238093 fps ================
=============5.59125105061644 fps =================
=============5.5699164860809764 fps ===============
=============5.564625486322171 fps ================
=============5.592979545900046 fps ================
=============5.614477716730373 fps ================
```

Enable Quantisation – quantize = True

```
(dlonedge) pi@pb004:~ $ python3 mobile_net.py
/home/pi/dlonedge/lib/python3.11/site-packages/torchvision/models/_utils
d' is deprecated since 0.13 and may be removed in the future, please use
  warnings.warn(
/home/pi/dlonedge/lib/python3.11/site-packages/torchvision/models/_utils
eight enum or `None` for 'weights' are deprecated since 0.13 and may be
 equivalent to passing `weights=MobileNet_V2_QuantizedWeights.IMAGENET1K
Net_V2_QuantizedWeights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
/home/pi/dlonedge/lib/python3.11/site-packages/torch/ao/quantization/uti
re calling calculate_qparams. Returning default values.
  warnings.warn(
/home/pi/dlonedge/lib/python3.11/site-packages/torch/_utils.py:410: User
e removed in the future and UntypedStorage will be the only storage clas
sing storages directly.  To access UntypedStorage directly, use tensor.u
  device=storage.device,
=============16.6219862041313 fps ================
=============19.91645571663891 fps ================
=============19.915476940864483 fps ===============
=============19.839643536678093 fps ===============
=============16.682428404444583 fps ===============
=============21.229794930185893 fps ===============
=============18.20355040128466 fps ================
=============15.924725625455313 fps ===============
=============21.704066783127715 fps ===============
=============19.221767325776405 fps ===============
=============20.60601829735328 fps ================
=============19.900476362187092 fps ===============
=============19.70807937448064 fps ================
=============19.64822666994894 fps ================
=============20.374951133178225 fps ===============
=============16.833020119541562 fps ===============
=============14.946884319595439 fps ===============
=============16.300363255630003 fps ===============
=============15.718592381026442 fps ===============
=============19.809916047750058 fps ===============
=============15.075895834205497 fps ===============
=============18.941769795479082 fps ===============
```

10 predictions

```
================================
52.99% monitor
33.47% screen
7.24% notebook
2.89% desktop computer
2.13% laptop
0.46% web site
0.21% oscilloscope
0.18% computer keyboard
0.12% iPod
0.09% menu
================================
40.88% monitor
25.82% screen
22.16% notebook
6.51% laptop
3.03% desktop computer
0.41% web site
0.36% iPod
0.26% menu
0.17% desk
0.14% computer keyboard
================================
36.93% monitor
31.69% screen
20.02% notebook
5.88% laptop
3.72% desktop computer
0.37% web site
0.28% desk
0.28% oscilloscope
0.24% computer keyboard
0.24% menu
================================
```



Quantization using Pytorch

Quantization aware training

Quantization-aware training (QAT) is the quantization method that typically results in the highest accuracy. With QAT, all weights and activations are "fake quantized" during both the forward and backward passes of training: that is, float values are rounded to mimic int8 values, but all computations are still done with floating point numbers.

```
[10] qnet = Net(q=True)
     fuse_modules(qnet)
     qnet.qconfig = torch.quantization.get_default_qat_qconfig('qnnpack')
     torch.quantization.prepare_qat(qnet, inplace=True)
     print('\n Conv1: After fusion and quantization \n\n', qnet.conv1)
     qnet=qnet.cuda()
     train(qnet, trainloader, cuda=True)
     qnet = qnet.cpu()
     torch.quantization.convert(qnet, inplace=True)
     print("Size of model after quantization")
     print_size_of_model(qnet)

     score = test(qnet, testloader, cuda=False)
     print('Accuracy of the fused and quantized network (trained quantized) on the test images: {}% - INT8'.format(score))
```

```
[5,   681]  loss 0.153281 (0.117172) train_acc 95.312500 (96.448627)
[5,   701]  loss 0.036884 (0.117355) train_acc 100.000000 (96.440289)
[5,   881]  loss 0.040150 (0.115852) train_acc 98.437500 (96.490715)
[5,   901]  loss 0.125532 (0.113059) train_acc 96.312500 (96.538568)
[6,     1]  loss 0.073349 (0.073349) train_acc 98.437500 (98.437500)
[6,   101]  loss 0.109321 (0.101780) train_acc 98.437500 (97.190876)
[6,   201]  loss 0.059297 (0.097198) train_acc 98.437500 (97.170398)
[6,   301]  loss 0.240343 (0.102414) train_acc 90.625000 (96.989203)
[6,   401]  loss 0.168531 (0.102163) train_acc 96.875000 (96.945137)
[6,   501]  loss 0.062470 (0.102711) train_acc 98.437500 (96.918663)
[6,   601]  loss 0.078745 (0.099992) train_acc 95.312500 (96.994502)
[6,   701]  loss 0.165914 (0.099554) train_acc 92.187500 (96.981999)
[6,   801]  loss 0.122891 (0.098583) train_acc 95.312500 (97.013409)
[6,   901]  loss 0.082342 (0.098820) train_acc 96.875000 (97.036279)
[7,     1]  loss 0.073371 (0.073371) train_acc 98.437500 (98.437500)
[7,   101]  loss 0.098834 (0.094540) train_acc 96.312500 (97.261757)
[7,   201]  loss 0.048159 (0.094441) train_acc 96.875000 (97.131539)
[7,   301]  loss 0.110617 (0.088093) train_acc 96.875000 (97.311047)
[7,   401]  loss 0.046228 (0.086850) train_acc 98.437500 (97.354271)
[7,   501]  loss 0.339986 (0.085287) train_acc 92.187500 (97.389596)
[7,   601]  loss 0.139136 (0.085410) train_acc 96.875000 (97.389767)
[7,   701]  loss 0.043594 (0.086802) train_acc 98.437500 (97.336394)
[7,   801]  loss 0.074818 (0.085420) train_acc 95.312500 (97.374378)
[7,   901]  loss 0.111683 (0.085265) train_acc 95.312500 (97.386774)
[8,     1]  loss 0.015970 (0.015970) train_acc 100.000000 (100.000000)
[8,   101]  loss 0.041880 (0.082745) train_acc 98.437500 (97.431031)
[8,   201]  loss 0.138863 (0.079846) train_acc 95.312500 (97.481343)
[8,   301]  loss 0.043438 (0.079473) train_acc 98.437500 (97.471068)
[8,   401]  loss 0.031117 (0.080110) train_acc 100.000000 (97.468441)
[8,   501]  loss 0.101823 (0.078971) train_acc 96.875000 (97.517465)
[8,   601]  loss 0.057876 (0.078848) train_acc 98.437500 (97.519759)
[8,   701]  loss 0.017625 (0.078419) train_acc 100.000000 (97.530230)
[8,   801]  loss 0.041831 (0.077788) train_acc 98.437500 (97.571395)
[8,   901]  loss 0.079823 (0.076922) train_acc 96.875000 (97.591218)
[9,     1]  loss 0.045722 (0.045722) train_acc 98.437500 (98.437500)
[9,   101]  loss 0.062270 (0.072831) train_acc 96.875000 (97.710396)
[9,   201]  loss 0.047946 (0.073389) train_acc 98.437500 (97.667918)
[9,   301]  loss 0.218342 (0.078462) train_acc 95.312500 (97.624058)
[9,   401]  loss 0.059447 (0.068210) train_acc 98.437500 (97.891989)
[9,   501]  loss 0.066881 (0.066582) train_acc 95.312500 (97.932268)
[9,   601]  loss 0.008523 (0.069176) train_acc 100.000000 (97.862937)
[9,   701]  loss 0.038815 (0.069375) train_acc 100.000000 (97.853513)
[9,   801]  loss 0.067438 (0.068579) train_acc 96.875000 (97.879694)
[9,   901]  loss 0.105698 (0.068196) train_acc 95.312500 (97.889498)
[10,     1]  loss 0.031723 (0.031723) train_acc 98.437500 (98.437500)
[10,   101]  loss 0.075317 (0.064380) train_acc 98.437500 (98.066213)
[10,   201]  loss 0.055871 (0.062318) train_acc 98.437500 (98.157649)
[10,   301]  loss 0.066978 (0.059288) train_acc 95.312500 (98.209095)
[10,   401]  loss 0.094356 (0.059530) train_acc 95.312500 (98.172537)
[10,   501]  loss 0.022216 (0.060803) train_acc 100.000000 (98.166168)
[10,   601]  loss 0.053476 (0.061582) train_acc 98.437500 (98.102121)
[10,   701]  loss 0.017416 (0.062274) train_acc 100.000000 (98.112972)
[10,   801]  loss 0.044746 (0.062803) train_acc 98.437500 (98.129292)
[10,   901]  loss 0.025757 (0.063854) train_acc 98.437500 (98.104537)
Finished Training
Size of model after quantization
Size (MB): 0.050984
Accuracy of the fused and quantized network (trained quantized) on the test images: 98.09% - INT8
```