

Summer Internship Report

Under the Guidance of Prof. Indrajit Chakrabarti (IIT KGP)

Submitted by

Kadagala Sai Siva Sankar (20EC39017)

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

Duration: 10th May 2024 – 20th July 2024

Problem statement: Deployable end-to-end optimized DNN framework for image classification

GitHub repository: [GitHub - Sai-SivaShankar/Summer_internship](#)

CONTENTS

- Acknowledgements
- Abstract
- Introduction
- Methodology
- Results
- Conclusion

Acknowledgements

I want to express my deepest gratitude to Prof. Indrajit Chakrabarti for his invaluable guidance and support throughout my internship. His insights and encouragement have been instrumental in shaping this work. I am also grateful to my PhD supervisor, Sayantan Dutta, who was assigned to me. His mentorship, patience, and expertise have been crucial to completing this internship.

Thank you both for your unwavering support and belief in my abilities.

Abstract

Deep learning models often require substantial computational resources, which limits their deployment on low-power devices like the Raspberry Pi. To address this, we utilize pruning and quantization techniques. Pruning reduces model complexity by eliminating less significant weights, enhancing inference speed and reducing memory usage. Quantization decreases parameter precision from 32-bit floating point to lower-bit integers, reducing model size and improving computational efficiency. We have developed an end-to-end optimized deep neural network (DNN) framework for image classification. This framework was tested on pre-trained models (LeNet5, ResNet50, MobileNet_v2, VGG16) using the MNIST dataset. For deployment, we employed a Raspberry Pi 4 Model B paired with a 12 MP RPI camera, utilizing the Picamera2 library for image capture, preprocessing, inference, and display. This optimized framework demonstrates the practical deployment of deep learning models on resource-constrained devices.

Introduction—

Deep learning models have become increasingly popular for various applications, including computer vision, natural language processing, and speech recognition. However, these models typically require extensive computational resources, making them challenging to run on low-power devices like the Raspberry Pi. One approach to addressing this challenge is to use pruning techniques to reduce the size of the deep learning models. Pruning involves removing less significant weights from the model, simplifying the network without substantially impacting its performance. This reduction in complexity can lead to faster inference times and lower memory usage, making the models more suitable for deployment on resource-constrained devices like the Raspberry Pi. Another effective approach is the quantization of models. Quantization reduces the precision of the numbers used to represent a model's parameters, typically from 32-bit floating point to 8/4/2-bit integers. This can significantly reduce the model size and improve computational efficiency without substantial loss of accuracy. Deploying complex deep learning models on devices with limited computational power and memory, such as the Raspberry Pi, becomes feasible by leveraging quantization. Pruning and quantizing deep learning models for the Raspberry Pi can help overcome low-power devices' computational and energy constraints, making it possible to run deep learning models in a broader range of devices.

Methodology:

For the UNet model, we have written the code explicitly using only the NumPy library. This code will be used as a supporting document for future projects.

Quantization is critical for enhancing performance, especially on resource-constrained devices like the Raspberry Pi. Two libraries have been used for this purpose: torch.quantization and optimum-quant. While the torch.quantization library supports weights to be quantized to int8 and float8 datatypes, and the optimum-quant library (a newer library) supports int8, int4, int2, and float8 datatypes. Depending on the model used and the quantization data type, we can use either post-training quantization or quantization-aware training techniques.

For pruning the model, we have used the torch-pruning library, which is based on structural pruning techniques. By specifying the desired channel pruning ratio, the pruner will scan all prunable groups, estimate weight importance, perform pruning, and fine-tune the remaining weights using our training code. This was further improved using a custom channel pruning ratio, although it requires understanding the model architecture.

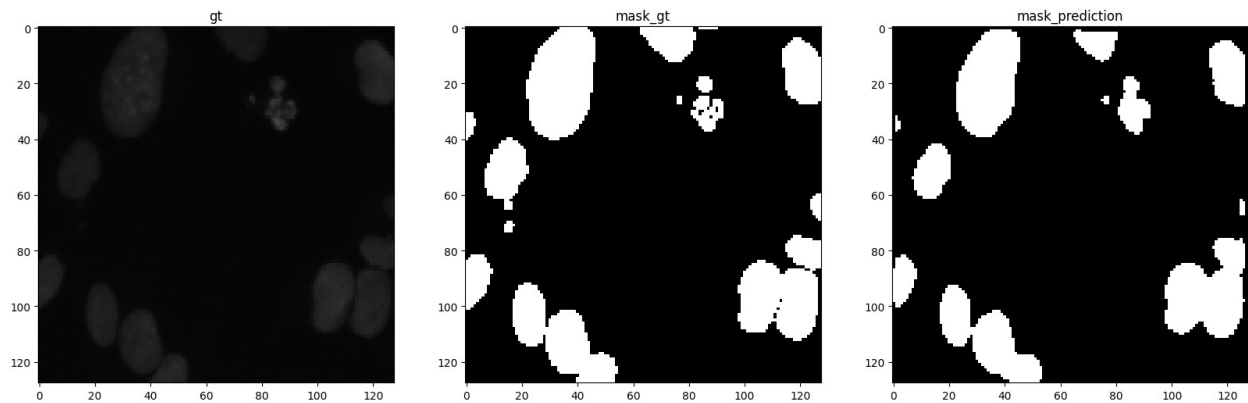
We created a pipeline consisting of pruning and quantization of the models. We tested this pipeline using pre-trained LeNet5, ResNet50, MobileNet_v2, and VGG16 models. The dataset we used is MNIST.

Finally, we used Raspberry Pi 4 model B and 12 MP RPI camera as our low-cost device setup for testing. We used the Picamera2 library to capture the frames using the camera, preprocess the image array, run inference on each frame and then display each frame with the class.

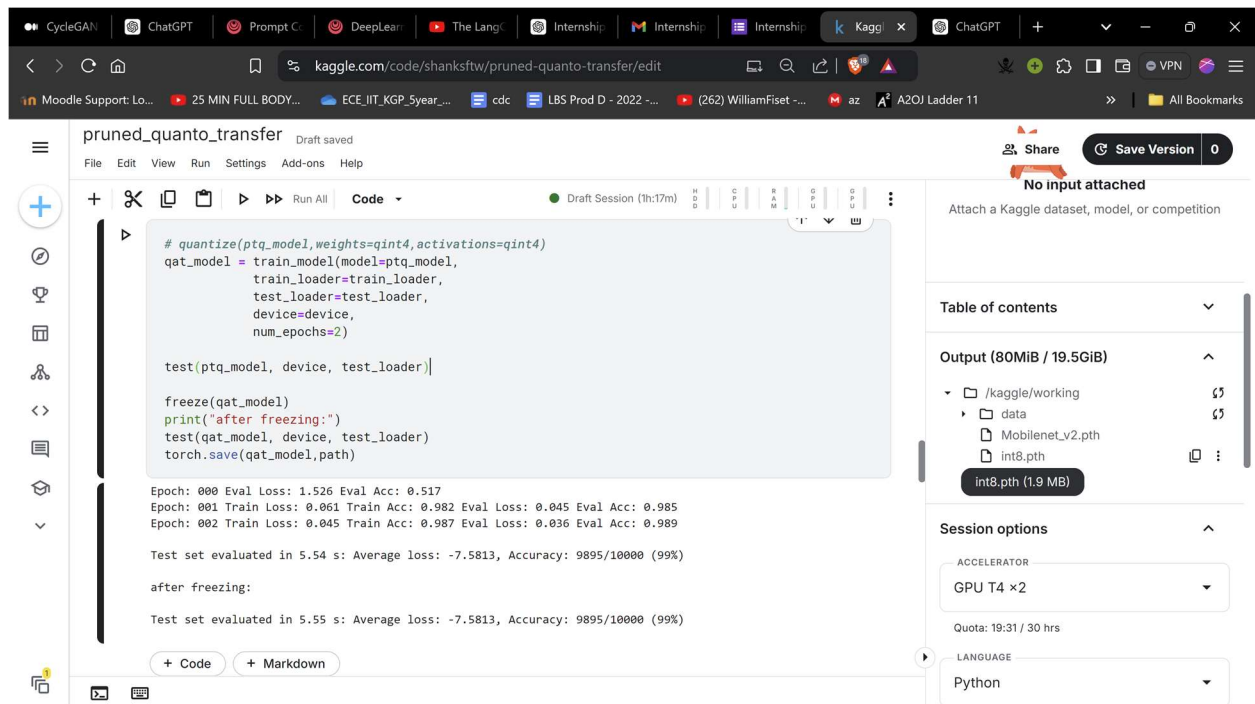


Figure 1: Deployable end-to-end optimized DNN framework

Results:



Segmentation mask obtained after training UNet



The screenshot shows a Kaggle Colab notebook titled 'pruned_quant_to_transfer'. The code in the notebook is as follows:

```
# quantize(ptq_model, weights=qint4, activations=qint4)
qat_model = train_model(model=ptq_model,
                        train_loader=train_loader,
                        test_loader=test_loader,
                        device=device,
                        num_epochs=2)

test(ptq_model, device, test_loader)

freeze(qat_model)
print("after freezing:")
test(qat_model, device, test_loader)
torch.save(qat_model, path)
```

The output of the notebook shows the following results:

```
Epoch: 000 Eval Loss: 1.526 Eval Acc: 0.517
Epoch: 001 Train Loss: 0.061 Train Acc: 0.982 Eval Loss: 0.045 Eval Acc: 0.985
Epoch: 002 Train Loss: 0.045 Train Acc: 0.987 Eval Loss: 0.036 Eval Acc: 0.989

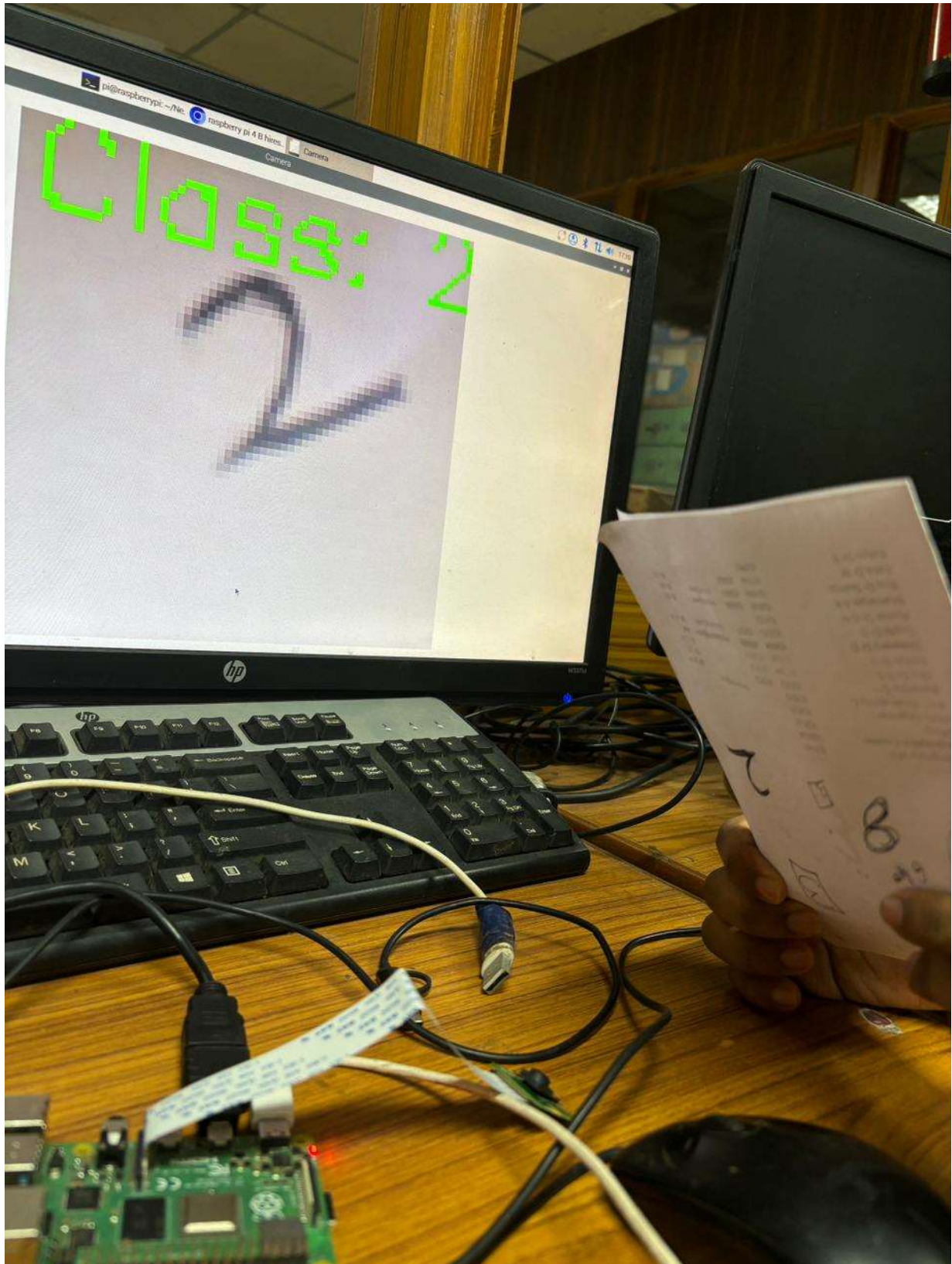
Test set evaluated in 5.54 s: Average loss: -7.5813, Accuracy: 9895/10000 (99%)

after freezing:

Test set evaluated in 5.55 s: Average loss: -7.5813, Accuracy: 9895/10000 (99%)
```

The right sidebar of the notebook shows the 'Table of contents' and 'Output (80MiB / 19.5GiB)' sections. The 'Output' section shows the file 'int8.pth (1.9 MB)'.

We have used the pipeline to compress the size of MobileNetV2 from 9.23 MB to 1.9 MB with almost equal accuracy.



Testing the optimised MobileNet_v2 after deploying it on Raspberry Pi 4

Conclusion:

Model accuracy can change significantly when deep image classification models are pruned. During pruning, careful tuning and a deep understanding of the model architecture are essential. One straightforward approach is to prune the fully connected layers, excluding the final classifier layer. It is observed that the accuracy of int8 quantized models is nearly equivalent to that of float32 models when using post-training quantization. However, int4 and int2 quantized models may require additional training, such as quantization-aware training, to maintain accuracy. This framework can be used in future projects to optimize any deep learning model.