**Agenda item:** 9.6

**Source:** Qualcomm Inc.

**Title: [FS\_AI4Media] Faster R-CNN Quantization**

**Document for** Discussion andAgreement

# Introduction

In this contribution, we provide results for the quantization of an object detection model, the Faster R-CNN.

# Model Quantization

The Faster R-CNN (Region-based Convolutional Neural Networks) is a well-known model for object detection tasks, which improves upon previous versions like R-CNN and Fast R-CNN. It was introduced by Ren and al. in their 2015 paper titled "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". Faster R-CNN addresses the efficiency issues of its predecessors by introducing a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

The Faster R-CNN architecture can be broadly divided into two main components:

1. Region Proposal Network (RPN):
	* The RPN is a fully convolutional network that predicts object bounds and detection scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are called anchors. These anchors are designed to be at multiple scales and aspect ratios to cover various object sizes and shapes.
	* The RPN takes an image (of any size) as input and outputs a set of rectangular object proposals, each associated with a score.
2. Detection Network:
	* The detection network takes the feature map generated by the shared convolutional layers and the region proposals from the RPN. Each proposal is then pooled into a fixed-size feature map and passed through a series of fully connected layers.
	* A SoftMax layer then classifies these regions into object categories or a background category, and bounding box regression is applied to predict the precise object location.

The entire system is a single, unified network for object detection that is trained end-to-end with a multi-task loss function that combines the losses of classification and bounding box regression.

The input to a Faster R-CNN model is an image or a batch of images. The images can be of different sizes, but they are often resized or padded to a fixed size to match the network's input dimensions for batch processing efficiency.

The outputs of a Faster R-CNN model for each input image include:

1. **Object Class Labels:** For each detected object, the model predicts a class label from a predefined list of categories (e.g., car, dog, person).
2. **Bounding Boxes:** For each detected object, the model outputs a bounding box that delineates the object's location within the image. These bounding boxes are defined by their coordinates (e.g., the top-left and bottom-right corners).
3. **Confidence Scores:** The model assigns a confidence score to each detected object, indicating the probability that the object belongs to a particular class.

Faster R-CNN significantly improved the speed and accuracy of object detection models, making it a foundational work in the field of computer vision. Its architecture has inspired many subsequent innovations and variations in object detection technology.

In its native form, the Faster R-CNN model is a floating-point model that has a size of about **173MB** for nearly 50 million parameters when using the ResNet-50 backbone.

To distribute this mode efficiently over networks, a sender-only quantization and pruning step may prove very helpful. In our experiment, we used the Neural Network Intelligence [1] (NNI) framework open-source software to perform quantization and pruning to reduce the model size to about **40MB**, with a quantization to int8. A fine-tuning step calibrated the quantized model weight using the Coco dataset to improve its prediction accuracy and precision. The original and the compressed model are uploaded to the GitHub repository.

It is worth noting that several attempts to compress other models, e.g. RetinaNet, have failed because most of these models were not traceable. That is, they contained custom code and python functions, which cannot be traced properly. The compression efficiency is adversely impacted when models are hard to trace.

The results are summarized in the following table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **person** | **car** | **truck** | **bus** | **bicycle** | **boat** |
|  | **Original** | **Quantized** | **Original** | **Quantized** | **Original** | **Quantized** | **Original** | **Quantized** | **Original** | **Quantized** | **Original** | **Quantized** |
| **Johnny** | 14.21 | 15.07 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BasketballDrill** | 12.74 | 13.16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BasketballDrive** | 9.75 | 9.17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BasketballPass** | 12.88 | 12.34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BlowingBubbles** | 50.85 | 31.39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BQMall** | 8.04 | 7.82 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **BQSquare** | 0.18 | 0.22 | 2.17 | 6.52 | 0 | 0 | 0 | 0 | 0 | 0 | 11.6 | 4.97 |
| **BQTerrace** | 2.99 | 3.26 | 24.07 | 28.56 | 32.35 | 27.28 | 38.88 | 27.78 | 0 | 0 | 0 | 0 |
| **Cactus** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **FourPeople** | 2.41 | 2.18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Kimono** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **KristenAndSara** | 8.11 | 6.71 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ParkScene** | 30 | 28.25 | 0 | 0 | 0 | 0 | 0 | 0 | 7.78 | 7.68 | 0 | 0 |
| **PartyScene** | 22.88 | 19.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **PeopleOnStreet** | 0.08 | 0.09 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **RaceHorses** | 2.59 | 2.68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Traffic** | 0 | 0 | 5.07 | 5.19 | 0 | 0 | 62.5 | 64.61 | 0 | 0 | 0 | 0 |

The table shows the mAP values for the different object classes. The mAP values are relatively low as the model was trained on a different dataset with different labels than what is used by SFU-HW-Objects.

It can be observed that the results for the quantized model are very comparable and, in most cases, superior to the original model.

# Proposal

We propose to add the content of section 2 to the evaluation TR.

# References

[1] Neural Network Intelligence (NNI), https://nni.readthedocs.io/en/latest/