Proposal: A Comparison of Optimization Methods for YOLO Running on Jetson Nano

Mark Jeiran and Igor Semyonov

George Mason University

March 20, 2024

1 Project Summary

YOLO is a highly capable image detection and classification model. It has been shown to perform well on various open datasets including coco128, CFAR10, CFAR100, and ImageNet. An edge device running YOLO could assist in various tasks such as human detection in home security cameras, quality assurance in manufacturing, and obstacle identification in autonomous systems. We propose to compare compression methods described in [2] to assess the performance of YOLOv8 on the Jetson Nano platform. We will write the harness in such a way as to enable easy assessment of deployment to other platforms as well.

2 ML ALgorithm

We will use YOLOv5 in this project. As mentioned earlier, You Only Look Once (YOLO) is a popular object detection model. The model is well known for its accuracy and efficiency. Figure 1 shows an overall structure of YOLO v5 as shown in Figure 1:

Backbone: YOLO v5 supports various backbone architectures. The backbone is a convolutional neural network (CNN), which acts as a feature extractor from the input image.

Detection Neck: Features extracted from the backbone are fed into the neck, which further process them. Feature fusion, dimensionality reduction, and spatial aggregation are often used to enhance the representation of features.

Detection Head: Detection head as the last layer is comprised of convolutional layers to predict bounding boxes and class probabilities. Typically, each predicted bounding box includes coordinates (x, y) for the box's center, width, and height, along with confidence scores for object presence and class probabilities.

3 Target Platform

Jetson Nano (16GB). NVIDIA Jetson Nano is a low-cost AI computer. It delivers the compute performance to run modern AI workloads at unprecedented size. It is incredibly

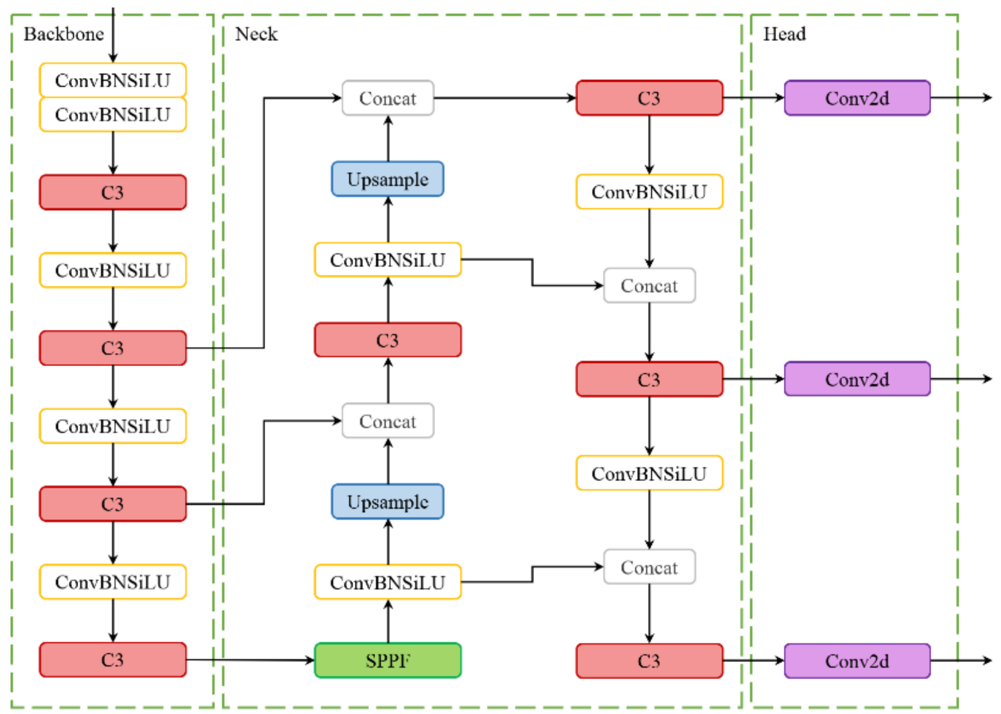

Figure 1: YOLOv5 architecture [2]

power-efficient, consuming as little as 5 watts.

4 Optimization Approach

We will train the model using a desktop workstation with a GPU. The model will be optimized using Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ) methods. QAT is the quantization of a pre-trained model then finetuning using training data to improve accuracy. The calibration process frequently occurs concurrently with the finetuning process in QAT. However, in PTQ a pre-trained model is calibrated using as a small subset of training data, to determine clipping ranges and scaling factors, then the model is quantized according to the calibration outcomes [2]. The primary difference in the methodology of these two methods is illustrated in Figure 2.

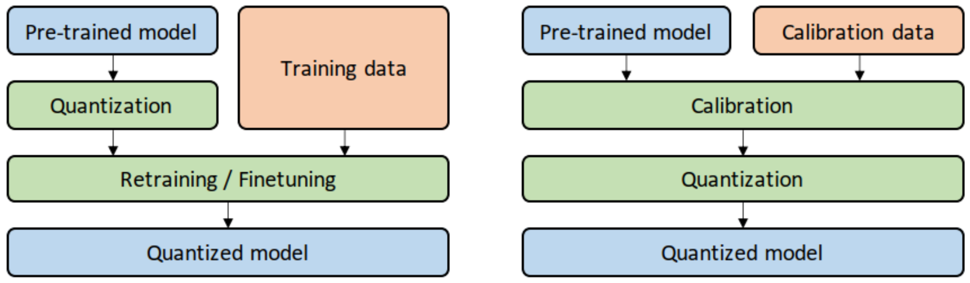

Figure 2: PTQ vs QAT quantization process comparison [1]

5 Metrics

We will evaluate the efficacy of our compression methods using GFLOPS vs Accuracy and number of Parameters vs Accuracy. We will also use the coco metrics of mean average precision (mAP) in both @0.5 and @0.95 variants as evaluation metrics.

6 Dataset

We will use CFAR100 as the train and test set. CIFAR100 has 60,000 32x32 color images in 100 classes, with 600 images per class. These images are split into 50,000 training images and 10,000 test images.

References

[1] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, *A survey of quantization methods for efficient neural network inference*, CoRR, abs/2103.13630 (2021).

[2] M. Jani, J. Fayyad, Y. Al-Younes, and H. Najjaran, *Model compression methods for yolov5: A review*, 2023.