

Improving the run-time performance of the object-detection model through pruning and quantization

Problem Statement

Deep neural networks (DNNs) have attracted a lot of attention in the recent years. The state-of-art accuracies of the deep neural networks come from their millions of parameters and billions of MAC operations which makes them difficult to deploy on edge-devices. Deploying DNNs on resource-constrained devices will open a dozen of possibilities for real-time applications in many fields. However, the use of the state-of-the-art models for real-time classification and object detection for real-time applications in resource-constrained devices is limited. In this work, the performances of the popular object detection will be accelerated using pruning and quantization techniques. Pruning is an important technique in which the unimportant parameters are removed from the network, and in quantization weights are stored with fewer bits.

Background

Object detection is a popular field in the deep learning has many real-time applications. There are number of object detection models proposed in the deep learning such as Fast RCNN, faster RCNN, SSD, YOLO etc. Out of the many object detection model available some are optimized for precision (mAP) and some are optimized for speed. Each object detection model has two parts that is detection of the object and classification. In object detection models, popular classification model are used as base classifier, VGG16, MobileNet, and ResNet are the popular choice among the many. Pruning is a popular way to make the model computational and power efficient by pruning filters which do not contribute much in reducing the error. Out of the various types of pruning, weight, node, layer, filter pruning is popular. Filter pruning doesn't make the pruned model unstructured and can be utilized directly on the end-device without the need of any special hardware or software.

Methodology

The whole process can be divided into 3 parts. First unimportant filters of the CNN will be pruned from convolutional layers. Next the weights of the resulting pruned model will be stored using reduced bit representation (8-bit quantization). Before applying the quantization, the model will be fine-tuned. In the third part the, the original vs enhanced models will be compared against various parameters such as mAP, storage, MAC operations, and energy consumption. Finally, the original and enhanced model will also be deployed on mobile devices for real-time performance comparisons. Figure 1 shows the overall methodology.

Step 1: Training the base models

The base model will be trained from scratch on the PASCAL VOC dataset. For this purpose any deep learning framework can be used like Keras, TensorFlow, PyTorch etc., However, PyTorch is preferred compare to other frameworks due to its flexibility. Pre-trained models can also be used for this purpose.

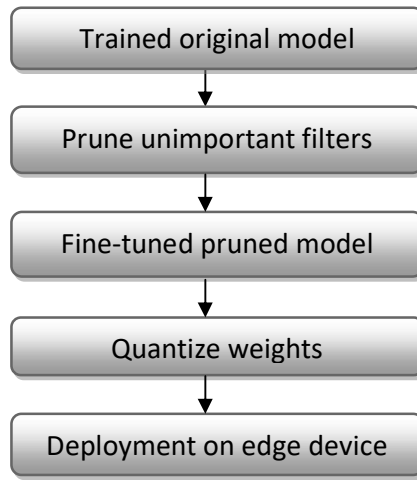


Fig. 1 Flowchart of the proposed approach

Step 2: Pruning filters

Once the base model is trained, now the important part is pruning the filters. It is important to decide the importance of the filters to be pruned from the convolutional layer, There are many criteria based on which the importance of the filter can be decided such their absolute sum, impact on the error etc. After pruning the filters a new model need to be crated and copy the weights from original model to new model of the remaining filters. There are several ways in which pruning can be applied, it can be applied on the original ImageNet trained classification model and the pruned models can be fine-tuned on the ImageNet dataset, However, training on ImageNet require adequate amount of computational resources, Secondly, the pre-trained can be fine-tuned using PASCAL VOC dataset.

Step 3: Fine-tune the pruned model

Pruning filters of the trained model degrades the performance of the model. In order to compensate the loss in the mAP due to the removal of the filters, a pruned model needs to be fine-tuned for more epochs. The process of pruning and fine-tuning can be iterative.

Step 4 Quantizing fine-tuned models

The weights of the DNNs are stored as 32-bit precision floating point numbers. However, it is found that the weights can be stored in reduced precision like 16-bit or 8-bit without significant loss in the model performance. To further improve the performances of the pruned and fine-tuned the model weights will be represented in 8-bit precision.

Step 4: Deploying on mobile device

The last part is the deployment of the model. After fine-tuning and quantizing, to compare the performance of the pruned models and effectiveness of the proposed approach, the pruned and original model will be deployed on real mobile-device. To deploy on the mobile TensorFlow lite or PyTorch can be used.

Experimental Design

Dataset and model

The first set of experiments will be performed on faster-RCNN and SSD-VGG16. After getting acceptable results, more efficient model will be taken into consideration for pruning such as SSD-Mobilenet, and YOLO. In the experiments PASCAL VOC dataset will be used to train and test the models.

Evaluation Measures

To compare the performance of the original and pruned models various evaluation measures will be used such as mAP (mean average precision), number of parameters in the original and the pruned model, number of FLOPs, and time taken to process the single image.

Software and Hardware Requirements

To large experiments on large models and datasets, GPU are required. In the absence of physical GPUs, there are many online platforms that could be utilized to conduct the experiments. It should also be noted that initial experiments can be performed on system with low computing power.

Input: Original trained model

Expected Output: An object detection model with improved performance

Mentor Name: Tejalal Choudhary