Real-time and accurate object detection on edge device with TensorFlow Lite

Junyan Dai

1 Student, High School Affiliated to Shanghai Jiao Tong University, Shanghai, 200434, China
*Corresponding author’s e-mail: daijunyan20@yeah.net

Abstract. Object detection is of vital importance to many fields, such as autonomous driving, outdoor robotics, and computer vision. Existing approaches on object detection can hardly run on the resource-constrained edge devices. In order to mitigate this dilemma, we propose to apply TensorFlow Lite to convert Float32 neural network model to uint8 neural network with subtle or even no accuracy loss. Two advantages are here for conversion. First, it reduces the model size to a quarter so that it fits for devices with limited storage. Second, it achieves much faster inference time. I conduct an experiment on MSCOCO dataset. Experimental results show that our proposed method achieves mAP 72.1 and FPS 23 on edge device.

1. Introduction
Object detection is a computer technology related to computer vision and image processing, which is used to detect instances of a class of semantic objects (such as people, buildings, or cars) in digital images and videos. Face detection and pedestrian detection are the research fields of target detection. Object detection is used in many fields of computer vision, including image retrieval and video surveillance. Apply this technique on edge device could achieve the goal such as autopilot. This paper will present how we speed up the detection progress with low accuracy lost on the edge device. So far, the state-of-the-art objection detection system is SSD (single shot MultiBox Detector)[1], it is a system based on the deep network but does not resample pixels or features for bounding box hypotheses with high accuracy. In the experiment based on MS COCO [4]Database, we test both Faster R-CNN and SSD to detect objection. The result is 58 FPS with mAP 72.1%(SSD300) or 23 FPS with mAP 75.1%(SSD500) vs. 7 FPS with 73.2% mAP (Faster R-CNN). We know from this experiment, SSD performs much faster with low accuracy lost. But it still could not completely operate on the edge device.

In order to make SSD run on smartphones, embed devices, or even edge devices smoothly and could detect in real-time, I convert this trained neural network from float data format to in format (float32 to uint8). This conversion has subtle, or even no accuracy lost with less storage size and faster inference time. My contributions is quantizing trained neural network SSD and make the run faster on the edge device with high accuracy and real-time object detection on the edge device (i.e., smartphone).

2. Related Works
(a) Related work in object detection
In the previous object detection[7] development, there are three approaches: R-CNN, Fast R-CNN[2][3], and Faster R-CNN[5]. All of them are two-stage approaches, proposal regions are first generated, and then CNN[6] classification is conducted. Faster R-CNN predominates the object detection because of the high precision in mAP but much faster inference time per image. These
approaches follow the same principle: hypothesize bounding boxes, resample pixels and features for each box, and the classifier. These three approaches all have high accuracy, but the massive amount of computation requirement makes the edge device hard to operate. In the experiment, although we used Faster R-CNN, the high-accuracy detector, it only operates 7 FPS. But we cannot deny that they have high accuracy, which has 73.2 mAP.

Although the previous R-CNN series has high accuracy, even after the development of Faster R-CNN, the detection of an image, as shown in the following figure, requires 7fps. In order to make the detection work can be used in real-time scenes, YOLO is proposed. YOLO is a one-stage approach that is the same as SSD in this paper. The one-stage approach is directly applied to the input image and the corresponding category and positioning are output. YOLO's approach of detection is different from that of the R-CNN series in that it solves the problem of target detection as a regression task.

(b) Related work in model quantization
Most convolutional neural network (CNNs) is not suitable for mobile devices because they mainly focus on classification/detection accuracy. The development of network architecture fails to take into account the complexity and computational efficiency of models. Successful deployment of CNNs on mobile platforms such as smartphones, AR/VR devices and drones requires small model sizes to accommodate limited device memory and low latency to maintain user engagement. This has led to an emerging field of research focused on reducing the model size and reasoning time of CNNs with minimal loss of precision. Methods in this area fall broadly into two categories.

In the first category, a new network architecture using efficient computing/memory operations is designed, such as MobileNet, SqueezNet, GostNet. The other one converts the weight and activation of CNN from a 32-bit floating-point to a lower bit-depth representation. The second approach of the quantization model will apply in this article.

(c) Related work in TensorFlow lite
Related Work: Briefly introduce TensorFlow history.
Evolution history. Since 2010, Google brains have created DistBelief as their first generation of proprietary machine learning systems. Google assigned computer scientists such as Jeffrey Sinton and Jeff Dean to simplify and restructure DistBelief's code base into a faster, more robust application-level code base, forming TensorFlow. In 2009, a research team led by Hinton significantly reduced the number of errors made using DistBelief's neural network, and made a scientific breakthrough in generalized back propagation through Hinton. Most notably, Hinton's breakthrough directly reduced errors in Google's speech-recognition software by at least 25 percent. [10]

In May 2017, Google announced that starting from Android Oreo, it would provide a software stack dedicated to Android development, TensorFlow Lite.

TensorFlow is a very popular machine learning and numeric computation framework. It is an end-to-end platform the builds on the static or dynamic graph. It excels at experimenting, designing, and deploying various machine learning algorithms.

TensorFlow has been applied in various international companies, such as Google, Uber, Microsoft, and other universities. TensorFlow Lite is specially designed for edge-based machine learning.

It enables us to run various light-weight algorithms on different resource-constrained edge devices, such as smartphones, micro-controller, and other chips.

TF Lite:
(a) Convert FP32 to UINT8 format.
Figure 1: This figure shows how to convert FP32 to UINT8 format.

3. **Real-Time Object Detection with SSDLite**

Object detection is very common in computer vision and image processing. It can detect a certain class of objects (such as people, vehicles) in the image and locate the object. Several approaches (R-CNN[6], Fast R-CNN[2][3], Faster R-CNN[5][12], SSD[1]) have been applied to many fields such as outdoor robotics, autonomous driving, and computer vision. The approach of object detection generally falls into hand-crafted feature-based or deep learning-based approaches. In this paper, I will discuss the approach based on deep learning.

Basically, to detect the object in the image by deep learning, we need the localization and classification.

The process of R-CNN (early object detector based on deep learning[9]):

- input an image and extract 1K-2K region proposal, then a segmentation method used to divide the image into very small areas. Look at the existing small areas and merge the two areas with the highest possibility. Repeat until the entire image is merged into a region position.
- To output all regions that have ever existed, so-called candidate regions. 
- Extract features of each proposal by CNN (convolutional neural networks)[8]

CNN-based methods are most widely used in object detection. Given an image, we divide it into several parts and assume each part as a single image. Then pass them through CNN and classify it into different classes. Finally, assigning each region to the corresponding category, we combined them to complete the target detection from the original image.

**Use regression to fine-tune the regional proposal position**

**What is SSD (single shot multibox detection)?**

Before SSD, there are three kinds of available approaches (R-CNN, Fast R-CNN, Faster R-CNN). All of them are slow, although the fastest one Faster R-CNN[5] can achieve 7FPS (detect seven images per second). So, it can hardly achieve real-time object detection. SSD is faster than the state-of-the-art single shot detector. SSD is the first object detector based on deep learning that does not resample the image pixels.

SSD is a one-stage detector, instead of two networks (region proposals Network and classifier network). In the process of sampling, different scales and aspect ratios can be adopted. Then, CNN can be used to extract features for direct classification and regression. The whole process only needs one step, so it runs much faster for detection in an image.

Traditionally, to detect different sizes of objects, which divides the image into different sizes and individuals treat each one. In SSD, we use different Convolution layer feature map can also achieve. The main network structure of the algorithm is VGG16. The two fully connected layers are changed to the convolution layer, and four more volume IOC product layers are added. The outputs of five different convolution layers are convolved with two convolution kernels of 3*3, one is confidence for
classification, and each default box generates 21 confidences. Localization (IOC) for output regression. Each default box generates four coordinate values (x, y, w, h). [11]

Figure 2: divide picture into different size

Feature map cell: it refers to each small cell in the feature map, which is each small square in the figure above. In the figure above, there are 64 feature map cells and 16 feature map cells respectively.

Default box refers to a series of boxes of fixed size on each feature map cell, namely the dotted box in the figure above, the blue square in (b). Ground truth means in machine learning, the data is labeled as <x, t>, t is ground truth. Just like in the figure above, x is the information of the box, and t is the information of the cat or the dog. Prior box: prior box refers to the actual selection of default box (actually, we do not take k default boxes of every feature map cell), the default box is a concept, the prior box is the actual selection. It is the red box in the (c).

An image is a feed to the network in training for each feature map, for a sample is training, will need the prior box with ground way box do matching (is an image input to the region in the network, determine the object area is the prior box, you can have a look at the fast RCNN), matching success shows that the prior box contains a goal, but from complete target ground way box and distance, The purpose of the training is to ensure the classification confidence of default box and return the prior box associated with ground truth box as tight as possible.

4. Model quantization.

Deep learning is effective in tasks such as Image Classification, Object Detection, and Natural Language Processing. Generally, the complexity of the model often limits its deployment in a variety of real-world application scenarios and resource-constrained edge devices. In order to apply SSD on the edge device, to eliminate the storage size and achieve faster inference time are significant. Quantization meanly is to convert the trained neural network from float 32 data format to unsigned int8 format (float32 to int8).

The formula of quantization is
\[ q = S r + Z \]
or
\[ r = S (q - Z) \]

where \( r \) is float 32, \( q \) is the quantized value, \( S \) is float32 scale, computed by involving min and max.

\[ S = \frac{r_{\text{max}} - r_{\text{min}}}{q_{\text{max}} - q_{\text{min}}} \]

\[ Z = \frac{q_{\text{max}} - r_{\text{max}}}{S} \]

(a) TensorFlow LITE

In TensorFlow, quantization is work with fake quantization node. Quantization-aware training. The training of the quantitative perception model ensures that the forward transfer matches the accuracy of the training and reasoning which shows in figure one. There are two aspects to this: Operator fusion in reasoning is precisely modeled during training the quantitative effects of reasoning are modeled during training.

In order to carry out effective reasoning, TensorFlow combines the batch normalization with the previous convolution and full connection layer by folding the batch norm layer and then quantifies it.

The quantization error is modeled by using a pseudo quantization node, and the quantization effect of forwarding and backward channels is simulated.

The forward transfer model is quantized, and the backward transfer model is quantized as a straight-through estimation. The quantization of weights and activities is simulated in both forward and backward transfers. In the process of backpropagation, parameters need to be updated with high
precision to ensure accuracy when accumulating minor adjustments of parameters. In addition, the minimum and maximum activation values are determined during training.

This allows models that have been quantitatively trained in the loop to be easily converted into fixed-point inference models, eliminating the need for a separate calibration step.

![Figure 3(a): Integer-arithmetic-only inference](image)

![Figure 3(b): Training with simulated quantization](image)

5. **Experiment and Discussion**

We comprehensively evaluate our method on the MSCOCO[4] dataset. This dataset consists of about 35 categories and 7.7 instance targets per image, less than 20% of images contain only one category, and only 10% of images contain only one instance target. The images in MSCOCO[4] contain natural pictures and common target pictures in life. The background is relatively complex, the number of targets is large, and the size of targets is smaller. Therefore, the task on COCO data set is more difficult.

(a) Mathematically introduce the metric to use to evaluate the performance. (IoU, precision and recall)

There are several metrics use to evaluate the performance of our works.

Intersection over Union (IOU): The IoU calculates the ratio of the intersection and union of the predicted and true borders.
Figure 4: Precision and recall.

Assuming that the current dataset has several images in two categories, our goal is to identify images in category A rather than category B. Now make the following definition:

True positives: The pictures are recognized correctly as A.
True negative: Pictures of B are not recognized, and the system rightly thinks that they are B.
False positives: Pictures of B were mistakenly identified as A.
False negatives: Pictures of A are not recognized, and the system mistakenly thinks that they are B.

\[
\text{precision} = \frac{tp}{tp+fp}, \quad \text{it means the accuracy of detection}
\]

\[
\text{recall} = \frac{tp}{tp+fn}, \quad \text{it means Recall rate.}
\]

Object detection

MobileNet can also be deployed as an effective infrastructure for working in modern target detection systems. We report the results of MobileNet training subjects to detect cocoa data based on recent work that won the 2016 Cocoa Challenge. In Table 1, MobileNet is compared with VGG and Inception V2 under the swift-RCNN and SSD frameworks. In our experiment, SSDS was evaluated using 300 input resolution (SSD 300), and faster-RCNN was compared with 300 and 600 input resolutions (faster-RCNN 300, faster-RCNN 600). The faster-RCNN model evaluates 300 RPN suggestion boxes for each image. Models trained on COCO train+ Val, not including 8K minimal images. Show the result in different pictures.

(3) Table 1 shows COCO object detection results in comparison using different frameworks and network architectures. mAP is reported with COCO primary challenge metric (AP at IoU=0.50:0.05:0.95)

| Framework Resolution | Model      | mAP  | Billion Mult-Adds | Million Parameters |
|-----------------------|------------|------|-------------------|--------------------|
| SSD 300               | deeplab-VGG| 21.1%| 34.9              | 33.1               |
|                       | Inception V2 | 22.0%| 3.8               | 13.7               |
|                       | MobileNet   | 19.3%| 1.2               | 6.8                |
| Faster-RCNN 300       | VGG         | 22.9%| 64.3              | 138.5              |
|                       | Inception V2 | 15.4%| 118.2             | 13.3               |
|                       | MobileNet   | 16.4%| 25.2              | 6.1                |
| Faster-RCNN 600       | VGG         | 25.7%| 149.6             | 138.5              |
|                       | Inception V2 | 21.9%| 129.6             | 13.3               |
|                       | MobileNet   | 19.8%| 30.5              | 6.1                |
Figure 5: objection detection results using MobileNet SSD.

This result shows that the SSD could run one edge device fast and accurately.

6. Conclusion

This paper shows the approaches to apply real-time object detection on edge devices with low inference time and high accuracy on depthwise separable convolutions. We introduce the Single-shot detection and TensorFlow Lite effects in the paper. The next step is to improve accuracy and lower the inference time.

Reference

[1] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. Computer Vision – ECCV 2016 Lecture Notes in Computer Science, 21-37. doi:10.1007/978-3-319-46448-0_2

[2] Girshick, R. (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2015.169

[3] Wang, X., Shrivastava, A., & Gupta, A. (2017). A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2017.324

[4] Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D. Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. Computer Vision – ECCV 2014 Lecture Notes in Computer Science, 740-755. doi:10.1007/978-3-319-10602-1_48
[5] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster-RCNN: Towards Real-Time Object Detection with Region Proposal Networks
[6] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2013.
[7] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016)
[8] Howard, A.G.: Some improvements on deep convolutional neural network based image classification. arXiv preprint arXiv:1312.5402 (2013)
[9] Simonyan,K.,Zisserman,A.:Very deep convolution a networks for large scale image recognition. In: NIPS. (2015)
[10] Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: ECCV 2012. (2012)
[11] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
[12] Zhang, L., Lin, L., Liang, X., He, K.: Is faster r-cnn doing well for pedestrian detection. In: ECCV. (2016)
[13] Bell, S., Zitnick, C.L., Bala, K., Girshick, R.: Inside outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: CVPR. (2016)