Quantization Mimic: Towards Very Tiny CNN for Object Detection

Pawan Prasad K (ME16B179), Gokulan R (CS15B033)

Introduction

The paper proposes a deep learning framework for object detection using very tiny CNN models. “Very Tiny” refers to the reduced number of filters in each convolutional layer. For example, Resnet18-1-16 refers to Resnet18 but with the number of filters in each layer reduced by 1/16. We try to perform object detection task using such a network. Advantages include very few parameters and very high speed in training and inference. The challenge lies in training the tiny network to detect accurately. The author proposes a method - “Quantization Mimic” for the same.

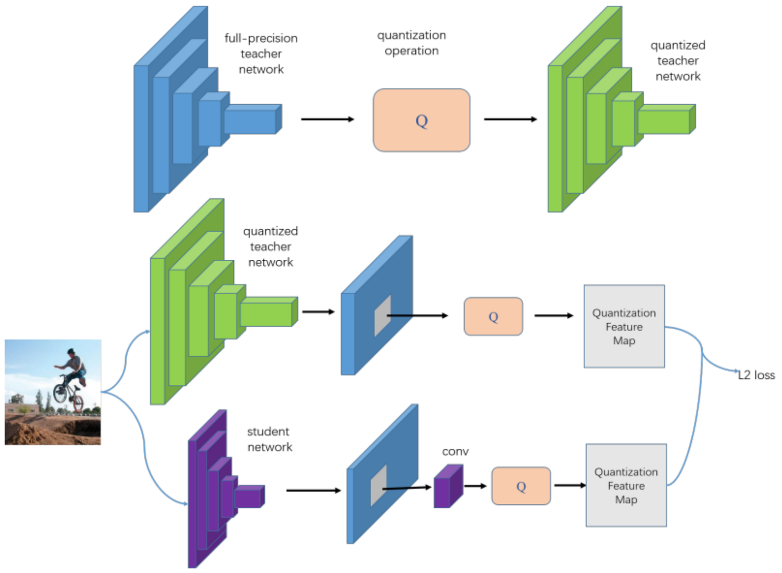
Background

Transfer Learning: In transfer learning, a pre-trained network (trained for a different dataset/classification classes) is reused and retrained to fit the new task. This technique of “transferring” the learned parameters is called transfer learning.

Quantization: Quantization reduces the number of unique elements in the model, thereby reducing the size of the model and speed up the inference. However, quantization is effective only on specialised accelerators.

Working

For quantization mimic, a large network (VGG16 or ResNet18) is chosen as a teacher network and a tiny network (VGG-1-16 or Resnet18-1-32) is chosen as a student network. A full-precision teacher network is first trained to maximum accuracy. Quantization is then performed on the teacher network so that the search space for the student network to learn is reduced. Now, training images are passed through both teacher and student networks, and loss calculated. The total loss is the sum of region proposal loss and object detection loss. In addition, a new loss is introduced, called mimic loss, which is a function of the feature maps generated by the teacher and student networks. All the losses are added, and this loss is backpropagated through the student network, by which the student network learns from the teacher network.



Experiments on Pascal VOC dataset

We perform object detection task using the VOC 2007 dataset consisting of 5011 training images. We use the Faster-RCNN framework on top of a Resnet18 CNN model for detection task [2]. mAP (Mean Average Precision) is used as the performance criteria for the following CNN frameworks coupled with Faster RCNN.

Resnet18 full precision

Convolutional layers of Resnet18 was used in Faster RCNN to generate filter maps. We used Stochastic Gradient Descent with a learning rate of 0.001, a momentum of 0.9 and weight decay of 0.0005. The training was run for 110k iterations. For each image, 300 proposals were considered. The *mean average precision (mAP) achieved was* ***67.64***, while it has been reported as ***72.9*** *in the paper*. Classwise mAP results are summarised in the table below.

class	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
AP	67.83	75.79	66.40	52.93	49.43	73.31	78.64	77.22	47.48	73.78
class	Dining table	dog	horse	motor bike	person	Potted plant	sheep	sofa	train	tvmonitor
AP	64.16	78.18	81.72	73.90	96.38	41.39	70.00	64.28	73.73	66.35

Resnet18 (Quantized)

For quantisation, we modify the **forward()** of the model. First, we get the output of the convolutional network (13 layers) and perform quantization on the **output feature maps**, and **not on any weights of the model**. This is done only during inference, and not during training. So, we reused the model obtained from the previous experiment. The mAP achieved was **67.64**, almost the same as the original version and the paper reports achieving mAP of **73.3**. However, the mAP values of individual classes differ slightly. Classwise mAP results are summarised below.

class	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
AP	67.93	75.39	67.21	53.35	49.31	74.19	78.65	76.83	47.05	71.17
class	Dining table	dog	horse	motor bike	person	Potted plant	sheep	sofa	train	tvmonitor
AP	64.75	78.79	81.68	74.17	76.65	41.07	70.30	64.61	73.55	66.25

Note: As a result of quantization, the features are converted to integer values. However, the

Resnet18-1-16

The code available in [2] creates a class **ResNet18** from Pytorch's inbuilt **torchvision** package. To construct ResNet18-1-16, we create a new class **ResNet18Student** in which we reduce the number of filters in each **conv** and **bn** layer by a **FACTOR**. The implementation was modularised such that this **FACTOR** is reconfigurable.

To construct an end-to-end network for detection, the code first takes the original ResNet architecture and adds a region-proposal network (RPN) as a hidden layer inside ResNet. The first 13 layers in ResNet18 generate the feature maps. Region proposals are drawn from this feature map, using the RPN. These proposals are then fed to the final five layers of Resnet, consisting of the last four **conv** layers and the final **fully-connected** layer, to generate detection scores (classification + bounding box). Additional modifications to other packages provided were done to train the new architecture.

Resnet18-1 -16 (Mimic)

To perform mimic, we need to calculate L2 loss between the feature maps of teacher and student networks. To do this, we reuse the full precision teacher network trained earlier. Given an image, we do a forward pass with the teacher network, and get bounding boxes of proposals. Next, we pass the image through the conv layers of ResNet (13 layers) and generate feature maps. This is done for both teacher and student network, and feature maps are stored separately. Now, for each bounding box, corresponding features from feature maps is extracted from both teacher and student feature maps, and L2 distance calculated. This value calculated is termed **mimic loss**. This error is backpropagated for the student network to learn the conv layer. After this, normal end-to-end training of the student network is done using Pascal VOC dataset, to train the Region Proposal Network (RPN) and detection network.

ResNet18-1-16 (Quantization and Quantization Mimic)

Quantization explained earlier can be modularly applied to the student network while training. Both methods (training from scratch and mimic) can use quantization. Quantization is applied only on the feature maps, and not on the weights. When quantization is applied, the outputs values are discrete and hence, the search space for training the student network reduces significantly. Hence, in this method of training, student network “mimics” the teacher network better, and produces better results.

References

1. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances In Neural Information Processing Systems (NIPS). (2015)
2. An easy implementation of Faster R-CNN: https://github.com/potterhsu/easy-faster-rcnn.pytorch
3. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. International journal of computer vision (IJCV) 88(2) (2010) 303–338
4. Code for Quantisation in Pytorch: https://colab.research.google.com/drive/1oDfcLRz2AIgsclkXJHj-5wMvbylr4Nxz#scrollTo=OgkWg605tE1y