CIS680: Vision & Learning
Assignment 3: MobileNet, ResNet, and Faster R-CNN
Due: Nov. 17, 2017 at 11:59 pm

Instructions

- This is an **individual** assignment. "**Individual**" means each student must hand in their **own** answers, and each student must write their **own** code in the homework. It is admissible for students to collaborate in solving problems. To help you actually learn the material, what you write down must be your own work, not copied from any other individual. You must also list the names of students (maximum two) you collaborated with.
- All the code should be written in Python. You are welcome to use Tensorflow or PyTorch to complete this homework.
- The CIFAR-10 dataset can be downloaded from [1]. In this homework, use train batch #1 for training and test batch for evaluation, each of which contains 1,000 images for 10 classes.
- The Transformed CIFAR-10 dataset for Faster R-CNN experiments can be downloaded from the course wiki website.
- You must submit your solutions online on **Canvas**. Compress your files into a ZIP file named "3_<penn_key>.zip", which should contain **1 PDF report** and **3 folders containing the Python code for each part**. Note that you should include all the figures in your report.

Overview

This homework aims at implementing and analyzing several well developed networks for feature extraction, localization and recognition. You will experiment and analyze three base networks, namely, ConvNet, MobileNet [4], and ResNet [3]. After developing the base networks, you can make use of their features to localize the object in an image. Finally, you will implement a simplified version of Faster R-CNN [2].

This homework consists of three parts.

1. Experiment and analyze three base networks, namely, ConvNet, MobileNet [4], and ResNet [3]. The analysis includes performance and computation efficiency.
2. Implement a region proposal network consisting of a proposal classifieer and regressor.
3. Implement Faster R-CNN [2], which includes a region proposal network and a object classifier.

Note that the full training of a network takes much time using only CPUs. You should observe the trend of training over the first couple hundreds of iterations and decide whether to finish training or not.

1 Base Networks: ConvNet, MobileNet, and ResNet (30%)

Before deep learning emerges, computer vision researchers design features to describe and/or discriminate images, e.g., HOG, SIFT, Gist, etc. Nowadays, researchers turn to design deep architectures to learn features that are useful for various tasks including object recognition, object detection, and semantic segmentation. In this part, you will build three base networks and compare their performance and efficiency using the CIFAR-10 dataset.

1. (5%) Train a network with architecture shown in Table 1.

   All convolution should preserve the input resolution with stride 1. Use Adam optimizer and exponential learning rate decay from 10^{-3} to 10^{-4} in $2,000$ iterations. Use weight decay 0.05.

   Report the final test accuracy and total training time. Count the total number of multiplications in convolution layers. (Do not count batch normalization.)

Layers	Hyper-parameters
Convolution 1	Kernel size = (5, 5, 32). Followed by BatchNorm and ReLU.
Pooling 1	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 2	Kernel size = (5, 5, 64). Followed by BatchNorm and ReLU.
Pooling 2	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 3	Kernel size = (5, 5, 128). Followed by BatchNorm and ReLU.
Pooling 3	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 4	Kernel size = (5, 5, 256). Followed by BatchNorm and ReLU.
Pooling 4	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 5	Kernel size = (3, 3, 512). Followed by BatchNorm and ReLU.
Pooling 5	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Softmax	(With fully connected layer) output channels = 10.

Table 1: Network architecture for part 1.

2. (10%) Replace the first 4 convolution layers with the separable convolution layers as shown in Figure 1.

   Tensorflow Guide: Use function tf.nn.depthwise_conv2d().

   Report the final test accuracy and total training time. Count the total number of multiplications in convolution layers. (Do not count batch normalization.)

3. (10%) Replace the first 4 convolution layers with the residual blocks as shown in Figure 2.

   Report the final test accuracy and total training time. Count the total number of multiplications in convolution layers. (Do not count batch normalization.)

4. (5%) Compare and explain the results in the previous questions.

2 Region Proposal Network (40%)

In this part, you will build a region proposal network step by step with the base networks you developed in the previous part. Region proposal network is a part of Faster R-CNN (Figure 3) for localizing objects in an image. You will implement a simplified version of region proposal network which consists of only one set of anchors (as opposed to nine sets

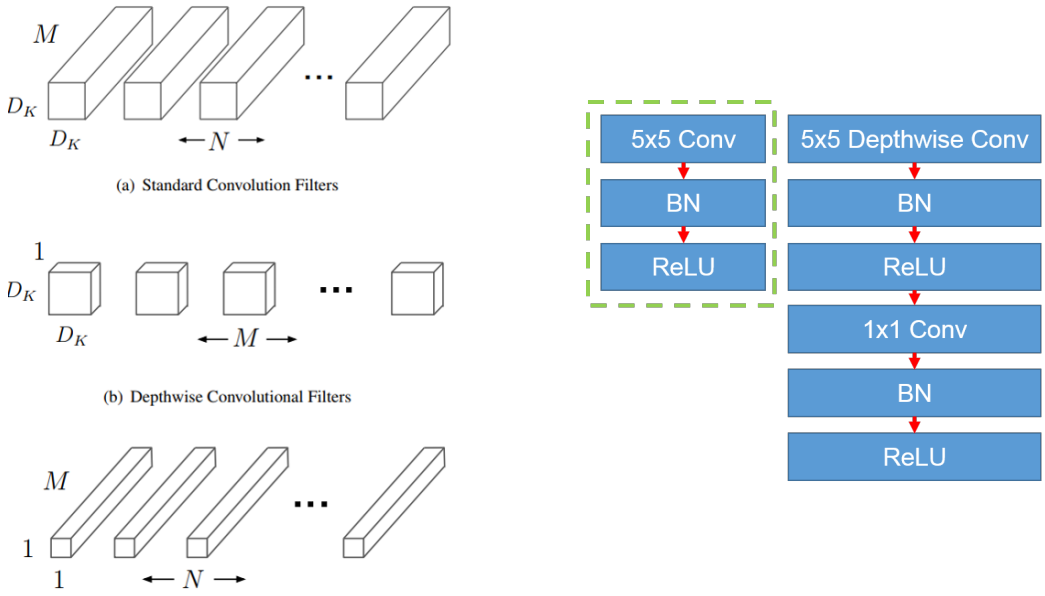

Figure 1: MobileNet and the separable convolution layer. A convolution layer (in dashed green box) can be converted to a separable convolution layer in the rightmost column.

in the paper). It's highly recommended to read the paper [2] beforehand. Starting from this part, use the Transformed CIFAR-10 dataset for experiments.

The Transformed CIFAR-10 dataset consists of images from the original CIFAR-10 dataset with random scaling from 0.5 to 2 and random shifting. The images with resolution $(48, 48)$ thus contain one object each with size from $(24, 24)$ to $(40, 40)$ in random. Note that the aspect ratio is always 1. Some sample images are shown in Figure 4.

Each image in the Transformed CIFAR-10 dataset comes with its label, location, and size. The file 'train.txt' and 'test.txt' consist of image names, class labels, row and column coordinates of the object centers, and widths of the objects in order.

To help you get started, each image also comes with a mask indicating the location of the object as shown in Figure 4. (The mask is blurred due to display.) The gray areas indicate the center of the object where the anchor box has over 0.7 Intersection-Over-Union (IOU) score; the black areas indicate less than 0.1 IOU score; The white areas should be ignored in training. The raw masks use 0's for black areas, 1's for gray areas, and 2's for white areas. The resolution of masks is $(6, 6)$ for the base network will down-sample the images by a factor of 8.

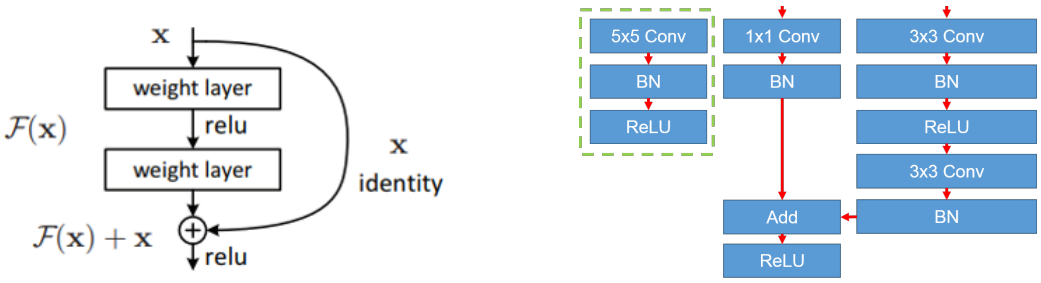

Figure 2: ResNet and the residual block. A convolution layer (in dashed green box) can be converted to a residual block in the rightmost column.

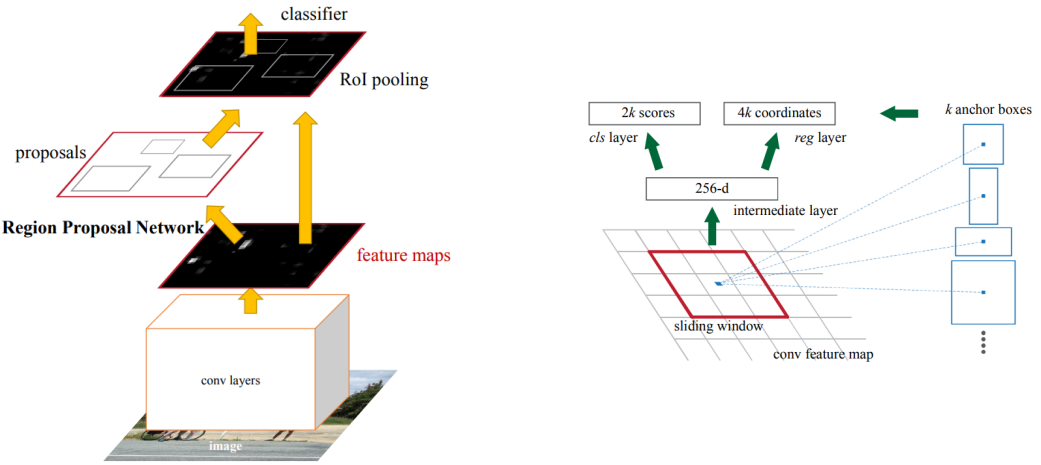

Figure 3: Faster R-CNN and its region proposal network.

1. (20%) Build a base network as shown in Table 2.

   Use the features (conv4) from the base network to build a proposal classifier (as the cls branch in Figure 3). Specifically, add a standard convolution layer (referred as the intermediate layer) with kernel size (3, 3, 256) (followed by BatchNorm and ReLU) and a convolution layer with kernel size (1, 1, 1) with no rectification layer. Use the ground truth masks and point-wise (i.e., over (6,6) feature map) sigmoid cross entropy loss to train the proposal classifier. The training procedure is the same as part 1.

   Plot the training loss over training iterations. Report the (point-wise) test accuracy of the proposal classifier.

   Tensorflow guide:

Figure 4: Images and masks in the Transformed CIFAR-10 dataset.

Some functions might be useful: tf.where(), tf.gather(), and

tf.nn.sigmoid.cross_entropy_with_logits().

Layers	**Hyper-parameters**
Convolution 1	Kernel size = (5, 5, 32). Followed by BatchNorm and ReLU.
Pooling 1	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 2	Kernel size = (5, 5, 64). Followed by BatchNorm and ReLU.
Pooling 2	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 3	Kernel size = (5, 5, 128). Followed by BatchNorm and ReLU.
Pooling 3	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 4	Kernel size = (3, 3, 256). Followed by BatchNorm and ReLU.

Table 2: Base network architecture for part 2.

2. (20%) In this question, you will build a proposal regressor (as the reg branch in Figure 3). On top of the intermediate layer, add another convolution layer with kernel size (1, 1, 3) (without BatchNorm and rectification). Note that you should initialize the biases to be (24, 24, 32) for each channel. The first two channels are for the row/column coordinates of the object center and the third channel for the width of the object.

   Parameterize the coordinates as follows:

$$t_x = (x - x_a)/w_a,\ t_y = (y - y_a)/w_a,\ t_w = \log(w/w_a)$$

$$t_x^* = (x^* - x_a)/w_a,\ t_y^* = (y^* - y_a)/w_a,\ t_w^* = \log(w^*/w_a)$$

where x, y, and w denote the box's center coordinates and its width. Variables x, x_a, and x^* are for the predicted box, anchor box, and groundtruth box respectively (likewise for y and w).

The regressor is trained with the Smooth L1 loss defined as:

$$L_{reg}(t_i, t_i^*) = \frac{1}{N_{reg}} \sum_i \text{smooth}_{L1}(t_i - t_i^*)$$

where

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

Note that the regressor loss is computed only on where the proposal masks equal to 1.

Train the whole network with two equally weighted losses from the proposal classifier and regressor. The training procedure is the same as the previous question.

Plot the training regression loss over training iterations. Report the final testing regression loss.

Tensorflow guide:

When the shape mismatch, consider functions such as tf.expand_dims(), tf.reshape(), tf.tile().

You will also need tf.where() and tf.gather() to ensure that you only impose regression loss on where the masks equaling 1.

tf.losses.huber_loss() can be used to model Smooth L1 loss.

3 Faster R-CNN (30%)

With the base network and region proposal network ready, you are now only one step from completing Faster R-CNN. After finishing the whole pipeline of Faster R-CNN, you can also change the base network to see the effect of different learned features.

1. (10%) One important layer that transforms the features of the proposal into the final object classifier is the ROI pooling layer. The ROI pooling layer warps the features of an ROI region into a fixed-sized feature map.

To save some effort, we provide a spatial transformer layer that can be used as ROI pooling layer. Check the usage in the Python file (spatial_transformer.py).

10% extra credits will be given if you program the ROI pooling layer without using spatial transformer layer or any other bilinear sampling layer.

The parameter θ to be fed into spatial transformer layer is as follows:

$$\theta[:,0] = w/48,\ \theta[:,1] = 0, \theta[:,2] = (x-24)/24,$$

$$\theta[:3] = 0,\ \theta[:,4] = w/48,\ \theta[:,5] = (y-24)/24$$

where x, y, w are the predicted box's column/row coordinates and width. Note that the predicted box is the maximally activated proposal.

Feed the original images into the spatial transformer layer with output resolution $(32, 32)$.

Display few outputs of the spatial transformer layer as shown in Figure 5.

Tensorflow guide: tf.argmax() and tf.gather_nd() might be useful.

2. (10%) Use the same θ parameter for the spatial transformer layer; however, feed in the feature map (conv4) from the base network and set the output size as $(4, 4)$.

   Add a fully convolution layer with 256 output channels (followed by BatchNorm and ReLU) and a 10-way Softmax layer (with fully connected layer) for classification.

   Train the network (consisting of proposal classifier, regressor, and object classifier) end-to-end with the same training procedure.

   Plot the 3 training losses over training iterations. Report the final classification test accuracy.

3. (10%) Replace the first three convolution layers in the base network (Table 2) with separable convolution layers or residual block respectively. Repeat the experiments in the previous question.

   Plot the 3 training losses over training iterations for two base networks. Report the final classification test accuracy. Compare the results of Faster R-CNN with all three base networks.

References

[1] CIFAR-10. https://www.cs.toronto.edu/ kriz/cifar.html.

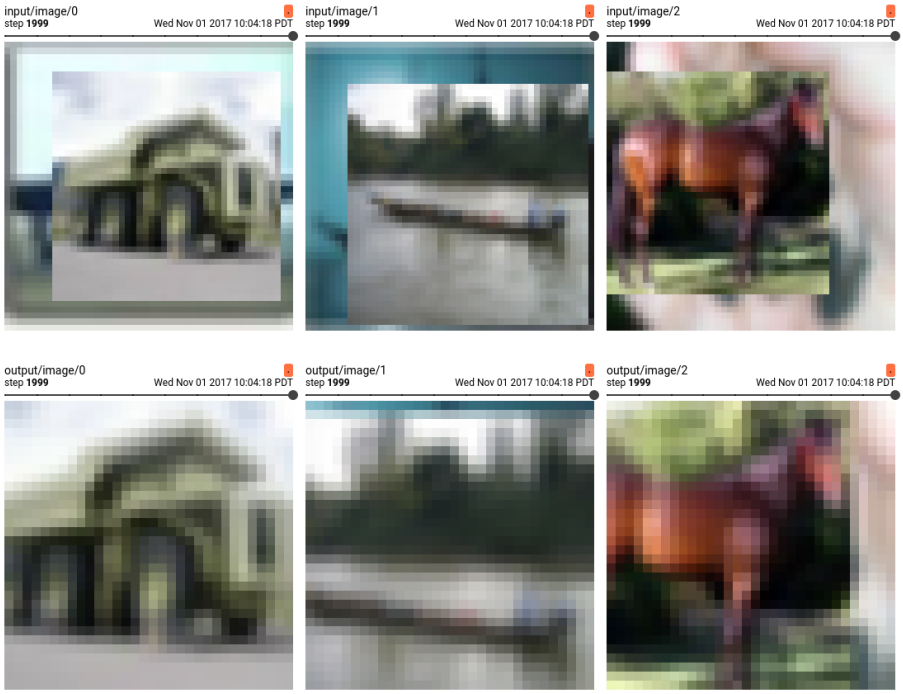

Figure 5: Inputs (top) and Outputs (bottom) of the spatial transformer layer with predicted proposal.

[2] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.