COMS E6998 section 012
PRACT DEEP LEARNING SYS PERF
Instructor: Parijat Dube
Due: October 31, 2021

Homework 3

Problem 1 - *Adaptive Learning Rate Methods, CIFAR-10* 20 points

We will consider five methods, AdaGrad, RMSProp, RMSProp+Nesterov, AdaDelta, Adam, and study their convergence using CIFAR-10 dataset. We will use multi-layer neural network model with two fully connected hidden layers with 1000 hidden units each and ReLU activation with minibatch size of 128.

1. Write the weight update equations for the five adaptive learning rate methods. Explain each term clearly. What are the hyperparameters in each policy ? Explain how AdaDelta and Adam are different from RMSProp. (5+1)
2. Train the neural network using all the five methods with L_2-regularization for 200 epochs each and plot the training loss vs number of epochs. Which method performs best (lowest training loss) ? (5)
3. Add dropout (probability 0.2 for input layer and 0.5 for hidden layers) and train the neural network again using all the five methods for 200 epochs. Compare the training loss with that in part 2. Which method performs the best ? For the five methods, compare their training time (to finish 200 epochs with dropout) to the training time in part 2 (to finish 200 epochs without dropout). (5)
4. Compare test accuracy of trained model for all the five methods from part 2 and part 3. Note that to calculate test accuracy of model trained using dropout you need to appropriately scale the weights (by the dropout probability). (4)

References:

- The CIFAR-10 Dataset.

Problem 2 - *Learning Rate, Batch Size, FashionMNIST* 30 points

Recall cyclical learning rate policy discussed in Lecture 4. The learning rate changes in cyclical manner between lr_{min} and lr_{max}, which are hyperparameters that need to be specified. For this problem you first need to read carefully the article referenced below as you will be making use of the code there (in Keras) and modifying it as needed. For those who want to work in Pytorch there are open source implementations of this policy available which you can easily search for and build over them. You will work with FashionMNIST dataset and MiniGoogLeNet (described in reference). If you cannot get MiniGoogleNet code from the reference you can do this question using LeNet.

1. Summarize FashionMNIST dataset, total dataset size, training set size, validation set size, number of classes, number of images per class. Show any 3 representative images from any 3 classes in the dataset. (3)
2. Fix batch size to 64 and start with 10 candidate learning rates between 10^{-9} and 10^{1} and train your model for 5 epochs. Plot the training loss as a function of learning rate. You should see a curve like Figure 3 in reference below. From that figure identify the values of lr_{min} and lr_{max}. (5)
3. Use the cyclical learning rate policy (with exponential decay) and train your network using batch size 64 and lr_{min} and lr_{max} values obtained in part 1. Here you will train till convergence and not just 5 epochs as in part 2. Plot train/validation loss and accuracy curve (similar to Figure 4 in reference). (5)
4. Fix learning rate to lr_{min} and train your network starting with batch size 64 and going upto 8192. Instead of training the network with different batch sizes for same number of epochs (as in part 2) you should train it for same number of iterations to get the training loss. You can keep the number of

iterations to train equal to the number of iterations when doing 5 epochs with a batch size of 64. If your GPU cannot handle large batch sizes, you can employ effective batch size approach as discussed in Lecture 3 to simulate large batches. Plot the training loss as a function of batch size. Do you see a similar behavior of training loss with respect to batch size as seen in part 2 with respect to learning rate ? (5)

5. Can you identify b_{min} and b_{max} from the figure in part 4 for devising a cyclical batch size policy ? Create an algorithm for automatically determining batch size and show its steps in a block diagram as in Figure 1 of reference. (4)

6. Use b_{min} and b_{max} values identified in part 3 and devise a cyclical batch size policy such that the batch size changes in a cyclical manner between b_{min} and b_{max}. In part 3 we did exponential decrease in learning rate as training progress. What should be an analogous trajectory for batch size as training progresses, exponential increase or decrease ? Use cyclical batch size policy (with appropriate trajectory) and train your network using learning rate lr_{min}. (6)

7. Compare the best accuracy from the two cyclical policies. Which policy gives you the best accuracy ? (2)

PS: In part 3 of problem we are doing cyclical learning rate with exponential decay. The code under "Keras Learning Rate Finder" in the blog implements triangular policy, you may need to change it to have exponential decay as mentioned in the first reference below. For part 4 and 6, you will be writing your own python project "Keras Batch Finder".

References:

1. Leslie N. Smith Cyclical Learning Rates for Training Neural Networks. Available at https://arxiv.org/abs/1506.01186.

2. Keras implementation of cyclical learning rate policy. Available at https://www.pyimagesearch.com/2019/08/05/keras-learning-rate-finder/.

Problem 3 - *Convolutional Neural Networks Architectures* 30 points

In this problem we will study and compare different convolutional neural network architectures. We will calculate number of parameters (weights, to be learned) and memory requirement of each network. We will also analyze inception modules and understand their design.

1. Calculate the number of parameters in Alexnet. You will have to show calculations for each layer and then sum it to obtain the total number of parameters in Alexnet. When calculating you will need to account for all the filters (size, strides, padding) at each layer. Look at Sec. 3.5 and Figure 2 in Alexnet paper (see reference). Points will only be given when explicit calculations are shown for each layer. (5)

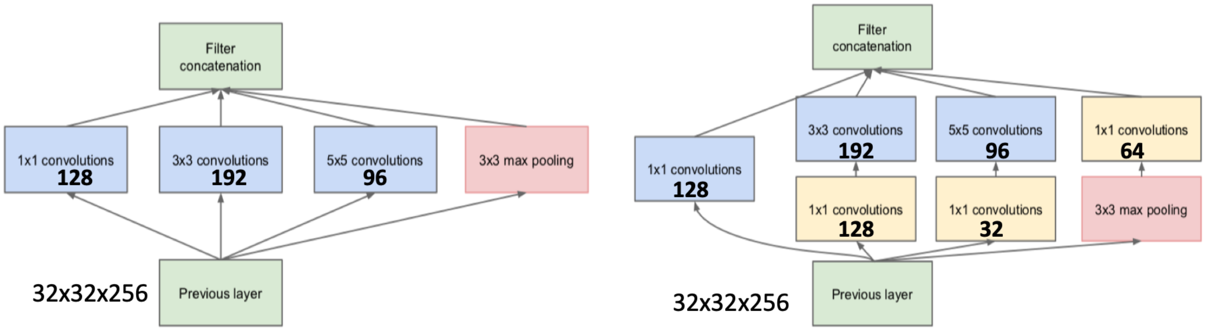
2. VGG (Simonyan et al.) has an extremely homogeneous architecture that only performs 3x3 convolutions with stride 1 and pad 1 and 2x2 max pooling with stride 2 (and no padding) from the beginning to the end. However VGGNet is very expensive to evaluate and uses a lot more memory and parameters. Refer to VGG19 architecture on page 3 in Table 1 of the paper by Simonyan et al. You need to complete Table 1 below for calculating activation units and parameters at each layer in VGG19 (without counting biases). Its been partially filled for you. (6)

3. VGG architectures have smaller filters but deeper networks compared to Alexnet (3x3 compared to 11x11 or 5x5). Show that a stack of N convolution layers each of filter size $F \times F$ has the same receptive field as one convolution layer with filter of size $(NF - N + 1) \times (NF - N + 1)$. Use this to calculate the receptive field of 3 filters of size 5x5. (4)

Layer	Number of Activations (Memory)	Parameters (Compute)
Input	224*224*3=150K	0
CONV3-64	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2	112*112*64=800K	0
CONV3-128		
CONV3-128		
POOL2	56*56*128=400K	0
CONV3-256		
CONV3-256	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256		
CONV3-256		
POOL2		0
CONV3-512	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512		
CONV3-512	28*28*512=400K	
CONV3-512		
POOL2		0
CONV3-512		
CONV3-512		
CONV3-512		
CONV3-512		
POOL2		0
FC	4096	
FC	4096	4096*4096 = 16,777,216
FC	1000	
TOTAL		

Table 1: VGG19 memory and weights

4. The original Googlenet paper (Szegedy et al.) proposes two architectures for Inception module, shown in Figure 2 on page 5 of the paper, referred to as naive and dimensionality reduction respectively.
   (a) What is the general idea behind designing an inception module (parallel convolutional filters of different sizes with a pooling followed by concatenation) in a convolutional neural network ? (3)
   (b) Assuming the input to inception module (referred to as ”previous layer” in Figure 2 of the paper) has size 32x32x256, calculate the output size after filter concatenation for the naive and dimensionality reduction inception architectures with number of filters given in Figure 1. (4)
   (c) Next calculate the total number of convolutional operations for each of the two inception architecture again assuming the input to the module has dimensions 32x32x256 and number of filters given in Figure 1. (4)
   (d) Based on the calculations in part (c) explain the problem with naive architecture and how dimensionality reduction architecture helps (*Hint: compare computational complexity*). How much is the computational saving ? (2+2)



(a) Inception module, naïve version

(b) Inception module with dimension reductions

Figure 1: Two types of inception module with number of filters and input size for calculation in Question 3.4(b) and 3.4(c).

References:

- (Alexnet) Alex Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. Paper available at https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- (VGG) Karen Simonyan et al. Very Deep Convolutional Networks for Large-scale Image Recognition. Paper available at https://arxiv.org/pdf/1409.1556.pdf
- (Googlenet) Christian Szegedy et al. Going deeper with convolutions. Paper available at https://arxiv.org/pdf/1409.4842.pdf

Problem 4 - *Batch Augmentation, Cutout Regularization* 20 points

In this problem we will be achieving large-batch SGD using batch augmentation techniques. In batch augmentation instances of samples within the same batch are generated with different data augmentations. Batch augmentation acts as a regularizer and an accelerator, increasing both generalization and performance scaling. One such augmentation scheme is using Cutout regularization, where additional samples are generated by occluding random portions of an image.

1. Explain cutout regularization and its advantages compared to simple dropout (as argued in the paper by DeVries et al) in your own words. Select any 2 images from CIFAR10 and show how does these images look after applying cutout. Use a square-shaped fixed size zero-mask to a random location of each image and generate its cutout version. Refer to the paper by DeVries et al (Section 3) and associated github repository. (2+4)
2. Using CIFAR10 datasest and Resnet-44 we will first apply simple data augmentation as in He et al. (look at Section 4.2 of He et al.) and train the model with batch size 64. Note that testing is always done with original images. Plot validation error vs number of training epochs. (4)
3. Next use cutout for data augmentation in Resnet-44 as in Hoffer et al. and train the model and use the same set-up in your experiments. Plot validation error vs number of epochs for different values of M (2,4,8,16,32) where M is the number of instances generated from an input sample after applying cutout M times effectively increasing the batch size to $M \cdot B$, where B is the original batch size (before applying cutout augmentation). You will obtain a figure similar to Figure 3(a) in the paper by Hoffer et al. Also compare the number of epochs and wallclock time to reach 94% accuracy for different values of M. Do not run any experiment for more than 100 epochs. If even after 100 epochs of training you did not achieve 94% then just report the accuracy you obtain and the corresponding wallclock time to train for 100 epochs. *Before attempting this question it is advisable to read paper by Hoffer et al. and especially Section 4.1.* (5+5)

You may reuse code from github repository associated with Hoffer et al. work for answering part 2 and 3 of this question.
References:

- DeVries et al. Improved Regularization of Convolutional Neural Networks with Cutout.
  Paper available at https://arxiv.org/pdf/1708.04552.pdf
  Code available at https://github.com/uoguelph-mlrg/Cutout
- Hoffer et al. Augment your batch: better training with larger batches. 2019
  Paper available at https://arxiv.org/pdf/1901.09335.pdf
  Code available at https://github.com/eladhoffer/convNet.pytorch/tree/master/models
- He et al. Deep residual learning for image recognition.
  Paper available at https://arxiv.org/abs/1512.03385