DS-UA 301
Advanced Topics in Data Science
Instructor: Parijat Dube
TAs: Astha Gupta, Rahul Sankar
Due: Oct 30, 2022

Homework 3

Problem 1 - *Learning Rate, Batch Size, FashionMNIST* 20 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 LeNet-5.

1. 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 for each learning rate. 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)
2. 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. Plot train/validation loss and accuracy curve (similar to Figure 4 in reference). (5)
3. We want to test if increasing batch size for a fixed learning rate has the same effect as decreasing learning rate for a fixed batch size. Fix learning rate to lr_{max} and train your network starting with batch size 32 and incrementally going upto 4096 (in increments of a factor of 2; like 32, 64...). You can choose a step size (in terms of number of epochs) to increment the batch size. Plot the training loss vs. $\log_2(batch\_size)$. Is the generalization of your final model similar or different than cyclical learning rate policy? (10)

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 2 - *Convolutional Neural Networks Architectures* 35 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. (4)
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

Homework 3

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

Homework 3

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. (3)

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 ? (2)

   (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. (3)

   (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. (3)

   (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)

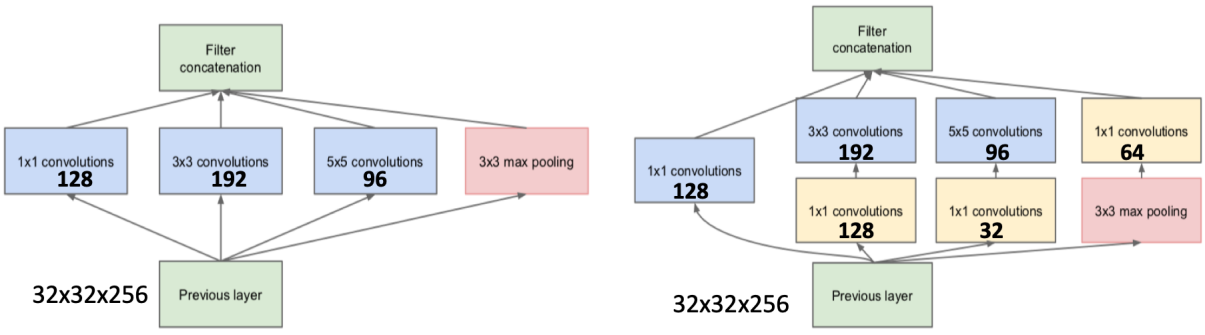
5. Faster-RCNN is a CNN based architecture for object detection which is much faster that Fast-RCNN. Read about Faster-RCNN in Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks and answer the following questions:

   (a) What is the main difference between Fast-RCNN and Faster-RCNN that resulted in faster detection using Faster-RCNN? (2)

   (b) What is Region Proposal Network (RPN)? Clearly explain its architecture. (2)

   (c) Explain how are region proposals generated from RPN using an example image.(3)

   (d) There is a lot of overlap in the region proposals generated by RPN. What technique is used in Faster-RCNN to reduce the number of proposals to roughly 2000? Explain how does this technique work using an example. (3)



(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:

Homework 3

- (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 3 - *Transfer learning: Shallow learning vs Finetuning, Pytorch* 25 points

In this problem we will train a convolutional neural network for image classification using transfer learning. Transfer learning involves training a base network from scratch on a very large dataset (e.g., Imagenet1K with 1.2 M images and 1K categories) and then using this base network either as a feature extractor or as an initialization network for target task. Thus two major transfer learning scenarios are as follows:

- *Finetuning the base model*: Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on Imagenet dataset. Rest of the training looks as usual however the learning rate schedule for transfer learning may be different.
- *Base model as fixed feature extractor:* Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

1. For fine-tuning you will select a target dataset from the Visual-Decathlon challenge. Their web site (link below) has several datasets which you can download. Select any one of the visual decathlon dataset and make it your target dataset for transfer learning. **Important : Do not select Imagenet1K as the target dataset.**
   (a) *Finetuning:* You will first load a pretrained model (Resnet50) and change the final fully connected layer output to the number of classes in the target dataset. Describe your target dataset features, number of classes and distribution of images per class (i.e., number of images per class). Show any 4 sample images (belonging to 2 different classes) from your target dataset. (2+2)
   (b) First finetune by setting the same value of hyperparameters (learning rate=0.001, momentum=0.9) for all the layers. Keep batch size of 64 and train for 50-60 epochs or until model converges well. You will use a multi-step learning rate schedule and decay by a factor of 0.1 ($\gamma = 0.1$ in the link below). You can choose steps at which you want to decay the learning rate but do 3 drops during the training. So the first drop will bring down the learning rate to 0.0001, second to 0.00001, third to 0.000001. For example, if training for 60 epochs, first drop can happen at epoch 15, second at epoch 30 and third at epoch 45. (6)
   (c) Next keeping all the hyperparameters (including multi-step learning rate schedule) same as before, change the learning rate to 0.01 and 0.1 uniformly for all the layers. This means keep all the layers at same learning rate. So you will be doing two experiments, one keeping learning rate of all layers at 0.01 and one with 0.1. Again finetune the model and report the final accuracy. How does the accuracy with the three learning rates compare ? Which learning rate gives you the best accuracy on the target dataset ? (6)

Homework 3

2. When using a pretrained model as feature extractor, all the layers of the network are frozen except the final layer. Thus except the last layer, none of the inner layers' gradients are updated during backward pass with the target dataset. Since gradients do not need to be computed for most of the network, this is faster than finetuning.

   (a) Now train only the last layer for 1, 0.1, 0.01, and 0.001 while keeping all the other hyperparameters and settings same as earlier for finetuning. Which learning rate gives you the best accuracy on the target dataset ? (6)

   (b) For your target dataset find the best final accuracy (across all the learning rates) from the two transfer learning approaches. Which approach and learning rate is the winner? Provide a plausible explanation to support your observation. (3)

For this problem the following resources will be helpful.

References

- Pytorch blog. Transfer Learning for Computer Vision Tutorial by S. Chilamkurthy
  Available at https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- Notes on Transfer Learning. CS231n Convolutional Neural Networks for Visual Recognition.
  Available at https://cs231n.github.io/transfer-learning/
- Visual Domain Decathlon

Problem 4 - *Weakly and Semi-Supervised Learning for Image Classification* 20 points

This problem is based on two papers, by Mahajan et al. on weakly supervised pretraining and by Yalinz et al. on semi-supervised learning for image classification. Both of these papers are from Facebook and used 1B images wiith hashtags. Read the two papers thoroughly and then answer the following questions. You can discuss these papers with your classmates if this helps in clarifying your doubts and improving your understanding. However no sharing of answers is permitted and all the questions should be answered individually in your own words.

1. Both the papers use the same 1B image dataset. However one does weakly supervised pretraining while the other does semi-supervised . What is the difference between weakly supervised and semi-supervised pretraining ? How do they use the same dataset to do two different types of pretraining ? Explain. (2)

2. These questions are based on the paper by Mahajan et al.

   (a) Are the model trained using hashtags robust against noise in the labels ? What experiments were done in the paper to study this and what was the finding ? Provide numbers from the paper to support your answer. (2)

   (b) Why is resampling of hashtag distribution important during pretraining for transfer learning ? (2)

3. These questions are based on the paper by Yalzin et al.

   (a) Why are there two models, a teacher and a student, and how does the student model leverages the teacher model ? Explain why teacher-student modeling is a type of *distillation* technique. (2+2)

Homework 3

(b) What are the parameters K and P in stage 2 of the approach where unlabeled images are assigned classes using teacher network ? What was the idea behind taking $P > 1$? Explain in your own words. (2+2)

(c) Explain how a new labeled dataset is created using unlabeled images ? Can an image in this new dataset belong to more than one class ? Explain. (2+2)

(d) Refer to Figure 5 in the paper. Why does the accuracy of the student model first improves as we increase the value of K and then decreases ? (2)

References

- Yalniz et al. Billion-scale semi-supervised learning for image classification.
Available at https://arxiv.org/pdf/1905.00546.pdf
- Mahajan et al. Exploring the Limits of Weakly Supervised Pretraining.
Available at https://arxiv.org/pdf/1805.00932.pdf