DS-UA 301
Advanced Topics in Data Science
Instructor: Parijat Dube

Homework 4

Due: November 5, 2023

General Instructions

This homework must be turned in on both Gradescope and Brightspace by 11:59 pm on the due date. It must be your own work and your own work only—you must not copy anyone's work, or allow anyone to copy yours. This extends to writing code. You may consult with others, but when you write up, you must do so alone. Your homework submission must be written and submitted using Jupyter Notebook (.ipynb). **No handwritten solutions will be accepted**. You should submit:

1. On **Gradescope**: one Jupyter Notebook(.ipynb) containing your solutions for each Problem in this assignment. Each Jupyter Notebook should be named as (*netid*)_hw(*#Homework*)_p(*#Problem*).ipynb.

   For example, *bz2058_hw4_p1.ipynb*.

2. On **Brightspace**: one .zip file containing all the notebooks, datasets, and other supporting materials involved in this homework. The .zip file should be named as (*netid*)_hw(*#Homework*).zip.

   For example, *bz2058_hw4.zip*.

Please make sure your answers are clearly structured in the Jupyter Notebooks:

1. Copy the Template Notebooks provided on Brightspace and write your solutions there.
2. Label each question part clearly. Do not include written answers as code comments. The code used to obtain the answer for each question part should accompany the written answer.
3. All plots should include informative axis labels and legends. All codes should be accompanied by informative comments. All output of the code should be retained.
4. Math formulas can be typesetted in Markdown in the same way as LaTeX. A Markdown Guide is provided on Brightspace for reference.

For more homework-related policies, please refer to the syllabus.

Problem 1 - *Convolutional Neural Networks Architectures* 40 points

In this problem, we will study and compare different convolutional neural network architectures. We will calculate the number of parameters (weights, to be learned) and the 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. You can refer to Section 3.5 and Figure 2 in the Alexnet paper referenced below. Points will only be given when explicit calculations are shown for each layer. (6)
2. VGG 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. You can refer to VGG19 architecture in Table 1 on page 3 of the paper by Simonyan et al referenced below. For this question, you need to complete Table 1 below for calculating activation units and parameters at each layer in VGG19 (without counting biases). It has been partially filled for you. (6)

Homework 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

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)

4. The original Googlenet paper by Szegedy et al. referenced below proposes two architectures for the 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 the inception module (referred to as "previous layer" in Figure 2 of the paper) has a size of 32x32x256, calculate the output size after each filter as well as filter concatenation for both the naive and dimensionality reduction inception architectures with the number of filters given in Figure 1. (4)

   (c) Next, calculate the number of convolutional operations for each of the two inception architectures again assuming the input to the module has dimensions 32x32x256 and the number of filters given in Figure 1. (4)

Homework 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)

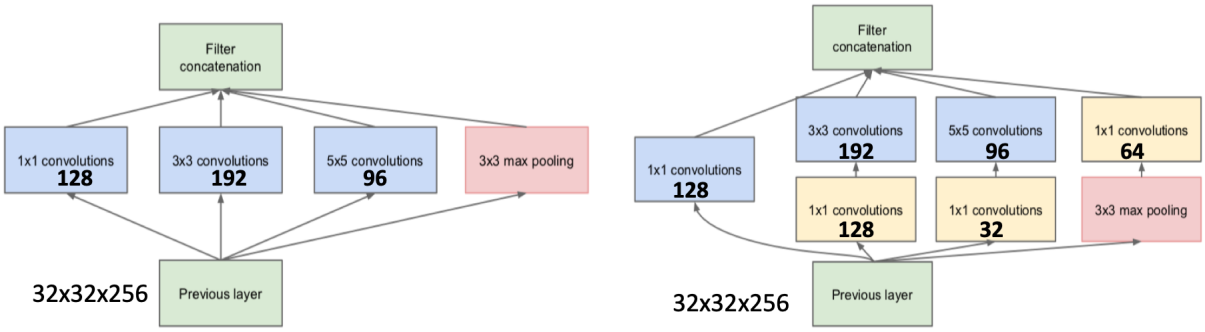

Figure 1: Two types of inception modules with the number of filters and input size for calculation in Problem 1.4(b) and 1.4(c).

5. Faster R-CNN is a CNN-based architecture for object detection which is much faster than Fast R-CNN. Read the Faster R-CNN paper referenced below and answer the following questions:

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

   (b) What is a 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 R-CNN to reduce the number of proposals to roughly 2000? Explain how this technique works using an example. (3)

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
- (Faster R-CNN) Shaoqing Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Paper available at https://arxiv.org/abs/1506.01497.pdf

Homework 4

Problem 2 - *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 a 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 pre-trained network, like the one that is trained on the Imagenet dataset. The 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 website (link below) has several datasets which you can download. Select any one of the visual decathlon datasets 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 pre-trained 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 the batch size of 64 and train for 50-60 epochs or until the 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, the first drop can happen at epoch 15, the second at epoch 30, and the third at epoch 45. (6)
   (c) Next, keeping all the hyperparameters (including the multi-step learning rate schedule) the same as before, change the learning rate to 0.01 and 0.1 uniformly for all the layers. This means keeping all the layers at the same learning rate. So you will be doing two experiments, one keeping the learning rate of all layers at 0.01 and one at 0.1. Again finetune the model and report the final accuracy. How does the accuracy of the three learning rates compare? Which learning rate gives you the best accuracy on the target dataset? (6)
2. When using a pre-trained model as a feature extractor, all the layers of the network are frozen except the final layer. Thus, except for the last layer, none of the inner layers' gradients are updated during the 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 at a learning rate of 1, 0.1, 0.01, and 0.001 while keeping all the other hyperparameters and settings the 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)

Homework 4

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
  Available at https://www.robots.ox.ac.uk/ vgg/decathlon/

Problem 3 - *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 with 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 models 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 leverage the teacher model? Explain why teacher-student modeling is a type of *distillation* technique. (2+2)
   (b) What are the parameters K and P in stage 2 of the approach where unlabeled images are assigned classes using the 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 improve as we increase the value of K and then decrease? (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

Homework 4

Problem 4 - *Siamese Network for Face Recognition* 15 points

In this problem, you will train a Siamese network in Pytorch and evaluate its performance with different contrastive loss functions. We also want to understand how robust are Siamese network for face recognition when presented with images of the same person, with and without glasses. You will work with this notebook which contains the code in the blog post referenced below.

1. Reuse the code in the notebook and follow the instructions to train and evaluate the Siamese network model in your own environment. (3)
2. The model outputs a dissimilarity score between two facial images. Using the training data come up with an appropriate threshold for a face recognition system that outputs a 0 if the two images are of the same person and 1 otherwise. (2)
3. Evaluate how your face recognition system performs when presented with two images of the same person, with and without glasses. You will use the MeGlass dataset as your test set. (3)
4. For robust face recognition, Mining-Contrastive loss is introduced in Section 3.2 of the paper by Guo et al referenced below. Train a new Siamese network using this loss function. Similar to part 2, use this network in face recognition by identifying the right threshold. (5)
5. Evaluate the performance of the new face recognition system obtained in part 4 on the test set of the MeGlass dataset. How does this compare with the performance of the old model used in part 3? (2)

References:

- Strahinja Stefanovic. Siamese Network in PyTorch with application to face similarity.
  Available at https://datahacker.rs/019-siamese-network-in-pytorch-with-application-to-face-similarity
- Guo et al. Face Synthesis for Eyeglass-Robust Face Recognition.
  Available at https://arxiv.org/pdf/1806.01196.pdf
- MeGlass dataset from the above paper
  Available at https://github.com/cleardusk/MeGlass