CSci 4695 and 6965: AI for Conservation
Fall Semester, 2023
Homework 3
Part 1 Due: Wednesday, September 27, 2023 at 11:59:59pm
Part 2 Due: Monday, October 2, 2023 at 11:59:59pm
Part 3 Due: Wednesday, October 4, 2023 at 11:59:59pm

Each part is work 50 points toward your homework grade.

Note: Updated Friday September 22 at 9:15 pm.

Overview

This homework explores the basics of neural network design and training, first using the Fashion-MNIST data and then working on data collected in 2015 in Nairobi National park, primarily of Masai giraffes and plains zebras. For the latter, your end goal will be to answer a basic question: what species are seen in an image? This could be zebras, giraffes, or both. (The images have been filtered so that there is at least one animal in each.) Your tasks will be divided into three parts: (1) exploration and refinement of the FashionMNIST model from Lecture 8, (2) zebra/giraffe data splitting and formation of a Dataset, and (3) training a classification network built on top of a pre-trained network ("back bone network"). These parts are due in stages, with dates shown above. I don't want students trying to do this all at once.

Importantly, instructions for accessing the CCI machines are shown below.

As a final preliminary note, I am aware that this is a first time implementing and training neural networks. You should not need to write a large amount of code, but it can be tricky to test and debug. Please do not feel that you have to struggle on your own. Ask questions on Submitty and seek help from fellow students. If necessary we will devote time in class to help with coding, training and debugging.

Part 1

Make changes to try to improve both the data exploration and the results of the newtork and training on the FashionMNIST data from Lecture 08. Start by adding the following to the Jupyter notebook distributed for class:

- Print statistics on the number of images of each class in the training and the test data.
- Keep track of the best model and weights from the test (validation) set. Output a message each time a new best model is found. This should be tested at the end of each epoch.
- Output statistics on the performance of the final best model on each class. This must at least be the per-class accuracy, but could be as sophisticated as a confusion matrix where row i, column j of the matrix is the number of times the i-th category is correct, but the top answer from the network is the j-th category.

Once this is done, explore some of the following to improve the overall accuracy on the test data set:

- Changes to the network itself: number of layers, depth of the convolution layers, etc.
- Changes to the optimization — learning rate, momentum, optimizer, weight decay, etc.

- Addition of dropout or batch normalization. (You may need to read about these.)
- Changes to the batch size and number of epochs.

There is clearly much more here than you can explore so be selective and explore what you reasonably can. Document your results briefly but careful. What is the best network and training that you discovered?

Submit your results as a Jupyter notebook with experiments embedded. (It may be a good idea to reduce the frequency of output so that you can more easily see what's going on. If you want to be somewhat adventurous, learn how to use the tqdm package.)

Part 2

Start by downloading and unzipping the file

`https://drive.google.com/drive/folders/164Vo2RwZ_w4wpqnU43oonDpt2daYRmzI`

The unzipped directory contains a CSV file and a collection of image files. Each line of the CSV contains three items: (a) the file name of an image, (b) a 0 or 1 indicating if there is at least one giraffe in the image, and (c) a 0 or 1 indicating if there is at least one zebra. This is your starting point.

Here are your tasks.

1. Write a function that takes just one argument — the name of the CSV file – and generates a split of the images into train (70%), validation (15%) and test sets (15%). Each split should have approximately the same proportion of giraffe images and zebra images (and images containing both), but the images should be randomly assigned to the splits otherwise. Create output that shows the split.
2. Write a subclass of the PyTorch `Dataset` class that implements the functionality to creates datasets for your train, validation and test splits. The class should include transformations to map your image into the appropriate format and to resize to the correct input size for your network.
3. Write code that tests your Dataset class by iterating through the three instances you created to show they are disjoint and include all images from the provided directory.
4. Write code to explore your three datasets, displaying statistics on the number of images of each species in the split and displaying 10 randomly selected images. This type of data exploration should be standard practice in your work.

Submit a single Jupyter notebook showing the implementation and results outlined above. If you are unsure of what to output, think about what would be brief, easy to follow and make it clear that your answers are correct.

Part 3

For Part 3, please implement and train a network to predict what species are shown in an image. In particular, the trained network should decide if an image shows zebras and decide if an image shows giraffes. These decisions should be made independently, allowing the possibility of an image having both zebras and giraffes. One way to do this might be to have a separate network for each species, but this does not allow learning of common information. Instead you can form a combined loss function, using BCEWithLogitsLoss https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

This is just a combination of the cross-entropy loss values for zebras and giraffes separately.

Rather than building a network from PyTorch primitives, please use a pre-trained network (architecture and weights) as a backbone and add a fully-connected network on top. Train only this fully-connected network. Use the Dataset class you created for Part 2.

Submit a Jupyter notebook that shows your neural network model (class), the training and testing functionality, and your final results. Please also show test images (at least 5 and maybe up to 10) that were classified correctly and images that were classified incorrectly, both for zebras and for giraffes. Doing this is important to help understand when and why the network succeeds and fails, and it should be a regular part of your work.

Setting up CCI Environment

Here are preliminary instructions for

1. You should have received an email from CCI about your account username and password. Please use these details to login into the CCI portal.
2. Once you have logged in for the first time and setup 2FA (`https://secure.cci.rpi.edu/client/`), you should have SSH access to the machine.
3. Here are the SSH details: `https://docs.cci.rpi.edu/landingpads/`. You will need to use your username and password for the SSH connection.
4. Once you have SSHed into the machine, please navigate to your copy in the **"AI for Conservation" (AIC1)** project and work there.
5. If you want to launch a Jupyter notebook instance, you will need to **port forward**. Please follow below instructions for the same:
   (a) Run on server:
   ```
   $ nohup jupyter notebook --ip=0.0.0.0 --port=8880 &
   ```
   (b) In your local machine, type:
   ```
   $ ssh -N -p 22 <uname>@<machinename> -L 127.0.0.1:8880:<machinename>:8880
   ```
   (c) Then in your local browser, type: `localhost:8880`
   (d) To get token ID, run below command on server terminal:
   ```
   $ jupyter notebook list
   ```
6. Any changes in your Jupyter notebook will be saved to your homework folder in the CCI machine.
7. To download a submittable copy of your codebase from the CCI machine, use SCP commands.
8. Once you are done with your session, run the below kill command on the server:
   ```
   $ jupyter notebook stop 8880
   ```

From Jay McGlothlin: Here is an example on how your students could connect for interactive use. On the NPL front end node NPLFEN01, a student would execute

```
salloc -t 60 –gres=gpu:1 srun –pty bash -i
```

to request 1 GPU for interactive use for 1 hour. Additional notes can be found at

`https://docs.cci.rpi.edu/Slurm/#interactive`

This will give each student their own GPU, but they will have to wait for it to become available.