Assignment 1: Image Classifier for Cats and Dogs

Deadline: September 20, 2018 at 11:59pm
Late Penalty: There will be a late penalty assessed of 10% of achievable grade per day late, up to a maximum of five days, after which the assigned grade will be zero. The first day of penalty occurs at the moment the assignment is late, and each subsequent penalty occurs exactly 24 hours later.

Learning Objectives

In this assignment, you will train a convolutional neural network to classify an image into one of two classes: "cat" or "dog". In the process, you will:

1. Understand at a high level the training loop for a machine learning model.
2. Understand the distinction between training, validation, and test data.
3. Understand the concepts of overfitting and underfitting.
4. Investigate how different hyperparameters, including the learning rate and batch size, affect the success of training.

What To Submit

Submit the two files specified below for this assignment in a single zip file with the file name `assign1-StudentID.zip` where StudentID is your utorid. The files to submit are:

- A PDF file `assignment1-writeup.pdf` containing your the plots required and answers to the written questions in this assignment. Graded questions are located in each section, indicated by the heading *Questions*. Please answer these questions and include them in your report under the appropriate section headings.
- The PyTorch model file (`model_xxx`) for the best model that you have trained in section 5.3, and 'xxx' indicates the parameters for that model.

1 Setting Up Your Environment

For this assignment, you will need to install **PyTorch**, a software framework for machine learning. We will be using PyTorch for the rest of the course assignments and project.

1.1 Install PyTorch

1. On a command line, activate the conda environment for this course first:
   `source activate mie324` (macOS/Linux) or `activate mie324` (Windows).
   You should see `(mie324)` At the beginning of the command line now.
2. Download PyTorch from `https://pytorch.org/` for your specific operating system, and NVIDIA CUDA library version. Use the corresponding installation command shown on the website for your system.

3. You can test that your system has installed PyTorch successfully by bringing up a Python interpreter (i.e. run `python` in command line while inside your `mie324_2018` conda environment) and running '`import torch`'.

1.2 Dataset

We will make use of some of the CIFAR-10 data set [2], which consists of colour images of size 32x32 pixels for 10 categories.

For this assignment, we will only be using the `cat` and `dog` categories. We have included code that automatically downloads the dataset the first time that the main script is run.

2 Assignment Code Structure

In this assignment, we are taking a top-down approach to understanding how to train a neural network for a binary (i.e. two class) classification task. You will **not** need to write any code to define the neural network and train the model yourself. Instead, we provide you with a configuration file that you will edit/change to specify parameters that dictate how the neural network is trained. The Python code we are using is based on the PyTorch CIFAR-10 tutorial [1], with a few modifications, that was demonstrated in Lecture 2.

You will explore how the each *hyperparameter* specified in the configuration file affects the training process, and the final performance (accuracy and speed) of the model. In general, a hyperparameter is a parameter that we set before the training process, and remains fixed. It is distinct from the weights/parameters that are being set by the training process itself. Some kinds of hyperparameters determine the overall *architecture* of the model such as the number of layers of the neural net, or the number of units on a layer. Other kinds of hyperparameters dictate the the behaviour of the optimizer (which is the code that tries to improve the quality of the model) such as the *learning rate*, the *batch size*, or the structure of the training *loss function*. In this assignment, the hyperparameters that we will be exploring are: (1) learning rate, (2) batch size, (3) number of training epochs.

A high level flowchart of how the various files in this assignment are related is illustrated in Figure 1.

Each of these files serve a different purpose:

1. `configuration.json`: This configuration file, in JSON format [3], lists the values for different hyperparameters that you will use to train a specific model.

2. `train.py`: This Python file takes in the configuration file and trains a neural network model on the given training and validation data. It produces two kinds of files: (1) Several CSV files to that gives the training/validation classification error as well as the loss function over time and (2) A model parameters file, which contains the weights of the neural network at the end of training. The script also prints out the total number of seconds elapsed to complete the training.

3. `plot_train.py`: This Python file takes in the configuration file and looks for the corresponding CSV that was generated from the `train.py` script and creates plots from the CSV files showing loss/error curves over the training epoch.

4. `test.py`: This Python file takes in the configuration file and looks for the corresponding model parameter file, and runs the model the testing data to produce: (1) Confusion Matrix,

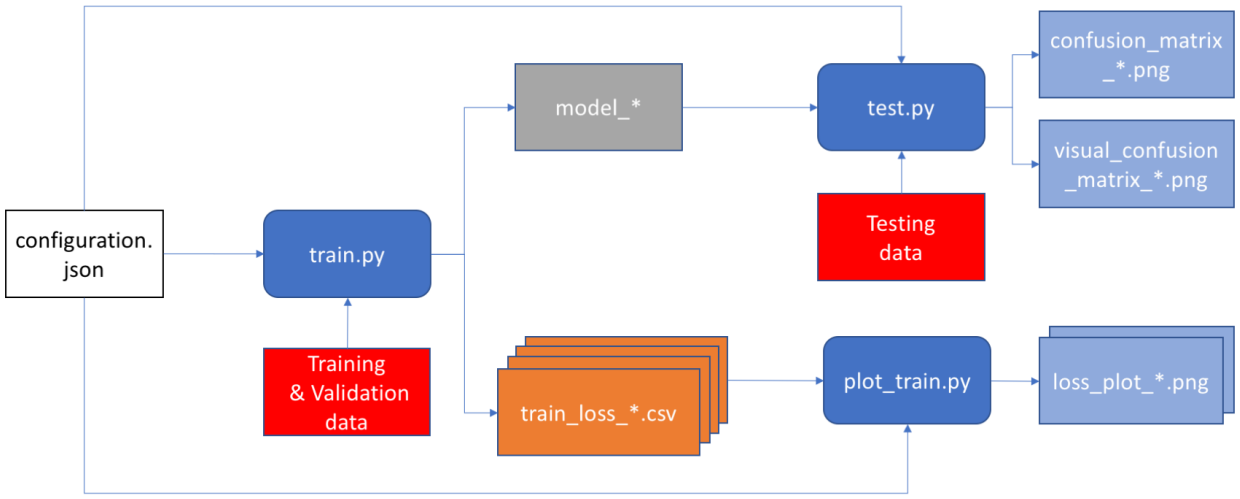

Figure 1: How the assignment's python scripts are structured

and (2) A "visual" confusion matrix. The confusion matrix shows the breakdown of how many examples were classified correctly/incorrectly for each class in the testing set. On the other hand, the "visual" confusion matrix has a similar table as the previous confusion matrix, but instead has an example image from the test set where the network's prediction was the most confident, whether correct or not.

All of these files were contained in the original zip file `assign1.zip` that you received this assignment in. Put all of these files into a separate folder.

3 Understand and Visualize the data (15 points)

To get an understanding of the kinds of images in the dataset, we will visualize the images in a mini batch of data. Launch PyCharm and open up a new project in the folder that you put the files in. Run the `visualize_data.py` script by right clicking on the `visualize_data.py` file in the project view (on the left hand side) and clicking **"Run 'visualize_data'"**. This will generate two image files: `example_cat.png` and `example_dog.png`. (Alternatively, to run the `visualize_data.py` script on the command line: use the command `python visualize_data.py` while in the folder with the code. You may find it very helpful to learn to use the command line environment.)

Questions

1. (2 points) Put the images `example_cat.png` and `example_dog.png`. that were output from the `visualize_data.py` script in your write up.

2. (3 points) How many training, validation, and test examples do we have for the combined "cat" and "dog" classes? (*Hint*: Take a look at the `get_data_loader` function in the `util.py` script. You may need to add a `print()` statement or use the debugger to determine the value of some variables).

3. (5 points) Why do we need a validation set when training our model? What happens if we

judge the performance of our models using the training set loss/error instead of the validation set loss/error?

4. (5 points) Why is it important to have equal number of training examples for the two classes? What can happen if there were significantly more training examples of one class over another?

4 Training the Network (15 points)

In this section, you will train the neural network model, saving the trained weights into a file. You will also store some statistics of the training progress into a CSV file. Figure 2 illustrates the high level steps in this process.

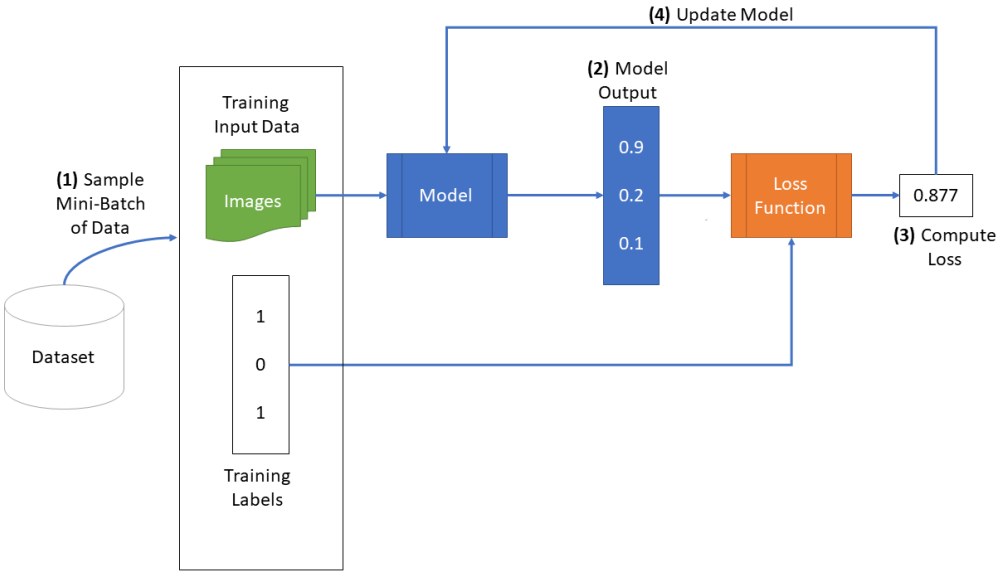

Figure 2: High level training loop for machine learning model

1. (3 points) Train the model using the **default** hyperparameters, specified in the original `configurations.json` by running the code `train.py`. This will generate a model file containing the weights, and also 4 .csv files. (A .csv file can be viewed as text, or opened in a spreadsheet program). The four .csv files give the values of the loss function (here we used cross entropy as the function, which hasn't yet been covered in class) and classification error rate for both the training data and the validation data over the training epochs. Notice that the name of the each of the five files (including the model file) contains the values of the three parameters from the configuration file - the number of epochs used, the batch size and the learning rate. **From the console output at the end of training, note the amount of time it takes to do this training in seconds, and report it in your document.**

2. In addition to the default hyperparameter values specified in the `configuration.json` file, we will explore how varying the values for each hyperparameter affects the learning process. For each of the following variations, modify the configuration file accordingly and run the `train.py` script as before, and keep the resulting .csv and model files.

(a) (6 points) Vary the **learning rate** to be {0.1, 0.001}[1], and use the default values for the rest of the hyperparameters.

(b) (6 points) Vary the **batch size** to be {32, 512}, and use the default values for the rest of the hyperparameters.

5 Evaluate the performance of the networks (70 points)

In this section, you will evaluate the performance of the models that you have trained in the previous section using quantitative metrics, and qualitative observations. (Recall that the trained models are now all stored in files generated by the training). You will pick the best model to do the final evaluation using the test set, and report the test set classification error.

5.1 Default Hyperparameter Settings (24 points)

First, make sure that your configuration file is reset to the default parameter values (i.e. the settings you used in section 4.1). Now run the code `plot_train.py`. This will produce two plots and put them into a .png file which you can view. The first plot is the value of the loss function for the training data and the validation data over the 50 epochs. The second one is the actual error rate (the fraction of incorrect results) for both sets of data over the 50 epochs. Based on these, answer the following questions for the evaluation of the model trained with the default hyperparameter value:

Questions

1. (10 points) Consider the plot of the **cross entropy loss** vs. epoch number for the train and validation set. There should be 2 curves (train and validation). What do you notice about the train vs validation curve as the training went on? Does this indicate underfitting, or overfitting? Why? Include the actual plots with your answer.

2. (10 points) Consider the plot of the **classification error** vs. epoch number for the train and validation set. There should be 2 curves (train and validation). What do you notice about the train vs validation curve as the training went on? How does the behaviour of the curves differ (if any) from the previous graph (cross entropy loss)? Include the actual plots with your answer.

3. (4 points) What is the final training/validation cross entropy loss and training/validation classification error at the end of training?

5.2 Evaluate the Hyperparameter Effects (24 points)

Now, for each of the hyperparameter settings in Section 4.2, run the `plot_train.py` code to generate the plots as before. We will compare the training curves between different hyperparameter settings.

Questions

[1]Note that {,} is the set notation to denote here that you will using the learning rate 0.1 and 0.001, not to define a range for you to pick a learning rate from.

1. (12 points) Include the plots of the **cross entropy loss** and **classification error** vs. epoch number for the train and validation set for the **3 learning rates (0.1, 0.01 (default), 0.001)**. There should be a total of 6 plots. How does the learning rate affect the loss and error curve? What happens when the learning rate is too high, as reflected by the graph? What happens when it is too low?

2. (12 points) Include the plots of the **cross entropy loss** and **classification error** vs. epoch number for the train and validation set for the **3 batch sizes (16, 64 (default), 512)**. There should be a total of 6 plots. How does the batch size affect the loss and error curve? What happens when the batch size is big, as reflected by the graph? What happens when it is very small? How is batch size related to the number of epoch that we train the network for?

5.3 Find the Best Hyperparameter Setting (10 points)

So far we have only tried modifying the learning rate and the batch size hyperparameter from the default value. But we have not changed the **number of epochs**. Since the model is saved only at the *end* after all the specified number of epochs have elapsed, this means that we can end up stopping the training too early (underfitting) or too late (overfitting).

For this section of the assignment, you will try to find the the best hyperparameter settings to train your network. A good start will be to simply pick the most promising setting seen so far and tune the number of epochs so that it has the best validation loss. In practice, to find the optimal set of hyperparameters we will need to modify several hyperparameters from the default value.

Questions

1. (4 points) What were the values of the 3 hyperparameters that performed best for you? Briefly describe how you found these hyperparameters.

2. (2 points) Include the plots of the **cross entropy loss** and **classification error** vs. epoch number for the train and validation set for the best hyperparameter. There should be a total of 2 plots.

3. (4 points) What is the final training/validation cross entropy loss and training/validation classification error at the end of training?

5.4 Evaluate the Best Model on the Test Set (12 points)

Set the best hyperparameters in the `configuration.json` file and then run the code `test.py`. This will plot confusion matrix and the "visual" confusion matrix. In the visual confusion matrix, each row will specify the true label, while the column indicates the network's output prediction. The "`prob`" value refers to the probability that it is a dog. For each (true label, model prediction) pair, we pick an image that had the highest confidence. Use the two plots to answer the following questions:

Questions

1. (6 points) Include the **confusion matrix** plot ("`confusion_matrix_*.png`"). How balanced is the confusion matrix? Is the model more likely to wrongly classify cats, or wrongly classify dogs?

2. (2 point) Given the numbers shown in the confusion matrix plot, compute the overall **test classification error**.

3. (4 points) Include the ***visual* confusion matrix** plot ("`visual_confusion_matrix_*.png`"). Comment on the image in each cell. Are the network's mistaken output plausible, or do the images clearly belong to their respective classes?

References

[1] Training a classifier. `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`. Accessed: 2018-07-15.

[2] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[3] Wikipedia. JSON — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=JSON&oldid=852198244`, 2018. [Online; accessed 30-July-2018].