COMP9444 Neural Networks and Deep Learning

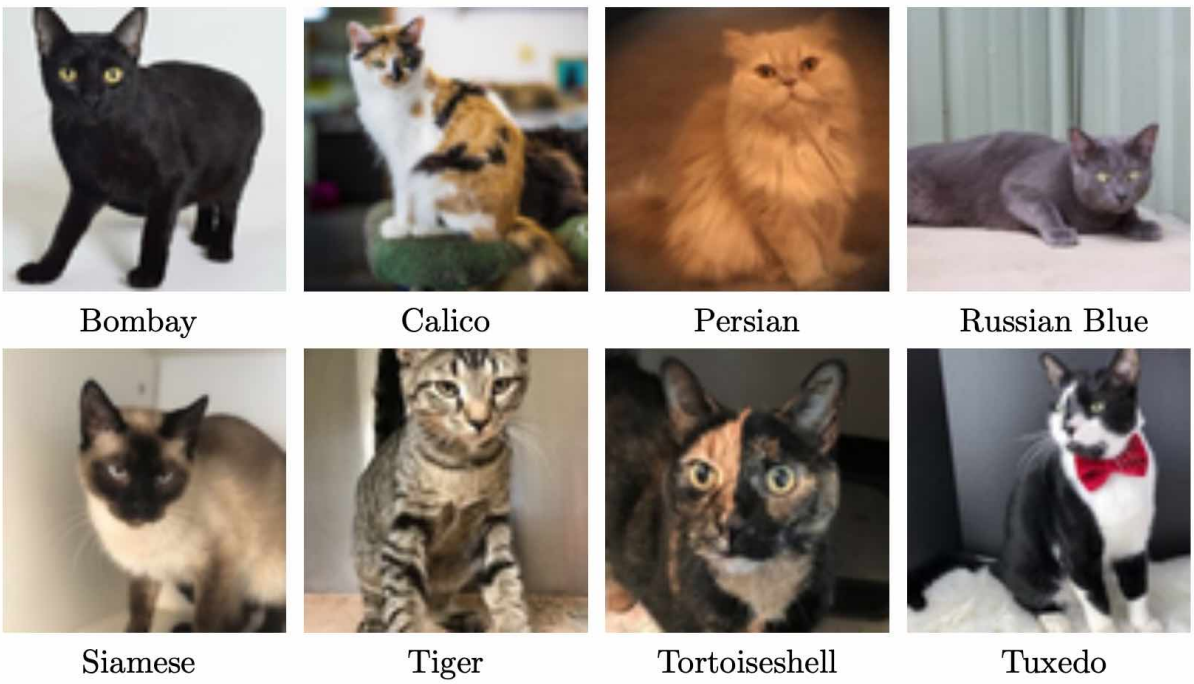
Term 3, 2021

Project 2 - Cat Breed Classification

Due: Friday 19 November, 23:59 pm
Marks: 30% of final assessment

Introduction

For this assignment you will be writing a Pytorch program which takes images of cats as input and learns to classify them into eight different breeds, using images that we provide.



Getting Started

Copy the archive `hw2.zip` into your own filespace and unzip it. This should create an `hw2` directory containing the main file `hw2main.py`, configuration file `config.py`, skeleton file `student.py` and data zip file `data.zip` . You will need to unzip the data file in the same directory to create a subdirectory named `data`. Your task is to complete the file `student.py` in such a way that it can be run in conjunction with `hw2main.py` by typing

```
python3 hw2main.py
```

You must NOT modify `hw2main.py` in any way. You should only submit `student.py`
(If you wish, you can modify `config.py` in order to switch between CPU and GPU usage)

The provided file `hw2main.py` handles the following:

- loading the images from the `data` directory
- splitting the data into training and validation sets (in the ratio specified by `train_val_split`)
- data transformation: images are loaded and converted to tensors; this allows the network to work with the data; you can optionally modify and add your own transformation steps, and you can specify different transformations for the training and testing phase if you wish

- loading the data using `DataLoader()` provided by pytorch with your specified `batch_size` in `student.py`

The training set contains 1000 images in each category, at resolution 80-by-80. You should aim to keep your code backend-agnostic in the sense that it can run on either a CPU or GPU. This is generally achieved using the `.to(device)` function. If you do not have access to a GPU, you should at least ensure that your code runs correctly on a CPU.

Please take some time to read through `hw2main.py` and understand what it does.

Constraints

We have tried to structure `hw2main.py` so as to allow as much flexibility as possible in the design of your `student.py` code. You are free to create additional variables, functions, classes, etc., so long as your code runs correctly with `hw2main.py` unmodified, and you are only using the approved packages (i.e. those available on the CSE machines). You must adhere to these constraints:

1. your model must be defined in a class called `Network`.
2. the `savedModel.pth` file you submit must be generated by the `student.py` file you submit.
3. make sure your version of pytorch and torchvision produce a `savedModel.pth` with the correct format to run on the CSE machines (which use `Torch1.8.1` and `TorchVision0.9.1`).
4. your submission (including `savedModel.pth`) must be under 50MB and you cannot load any external assets in the network class.
5. while you may train on a GPU, you must ensure your model is able to be evaluated on a CPU.

You must ensure that we can load your code and test it. This will involve importing your `student.py` file, creating an instance of your `Network` class, restoring the parameters from your `savedModel.pth`, loading our own test dataset, processing according to what you specified in your `student.py` file, and calculating the accuracy.

You may NOT download or load data other than what we have provided. If we find your submitted model has been trained on external data you will receive zero marks for the assignment.

Question

At the top of your code, in a block of comments, you must provide a brief answer (about 300-500 words) to this Question:

> Briefly describe how your program works, and explain any design and training decisions you made along the way.

You should try to cover the following points in your Answer:

a. choice of architecture, algorithms and enhancements (if any)
b. choice of loss function and optimiser
c. choice of image transformations
d. tuning of metaparameters
e. use of validation set, and any other steps taken to improve generalization and avoid overfitting

If you wish to refer to any graphs or figures in your Answer, you can put these in a file called `hw2.pdf`, and include it in your submission.

Groups

This assignment may be done individually, or in groups of two students. Groups are determined by an SMS field called hw2group. Every student has initially been assigned a unique hw2group which is "h" followed by their studentID number, e.g. `h1234567`. If you plan to complete the assignment individually, you don't need to do anything (but, if you do create a group with only you as a member, that's ok too). If you wish to form a

group, go to the COMP9444 WebCMS Page and click on "Groups" in the left hand column, then click "Create". Enter your Group Name and select the Group Type "hw2". After creating a Group, click "Edit", search for the other member, and click "Add". WebCMS assigns a unique group ID to each group, in the form of "g" followed by six digits (e.g. `g012345`). We will periodically run a script to load these values into SMS. You must ensure there are no more than two members in your group, and no-one is a member of two different groups.

Submission

You should submit your trained model and Python code by typing

```
give cs9444 hw2 student.py savedModel.pth
```

You must submit your trained model `savedModel.pth` as well as the Python code `student.py`

If you wish to include a supplementary file `hw2.pdf`, you can do this by typing

```
give cs9444 hw2 student.py savedModel.pth hw2.pdf
```

In order to avoid technical problems, you are strongly advised to submit from the command line, and not via the give Web interface. You can submit as many times as you like - later submissions by either group member will overwrite previous submissions by either group member. You can check that your submission has been received by using the following command:

```
9444 classrun -check hw2
```

The submission deadline is Friday 19 November, 23:59. 15% penalty will be applied to the (maximum) mark for every 24 hours late after the deadline.

Additional information may be found in the FAQ and will be considered as part of the specification for the project. You should check this page regularly.

When you submit, the system will check that your model can be successfully loaded, and evaluate it on data randomly chosen from a third dataset (disjoint from `data.zip` and also disjoint from the holdout test set) in batches of size 25. The reported accuracy might vary slightly each time you submit, due to changes in the randomly chosen subsets. The final holdout test set will be considerably larger and should give a stable evaluation.

Marking Scheme

After submissions have closed, your code wil be run on a holdout test set (i.e. a set of images and labels that we do not make available to you, but which we will use to test your model). Marks will be allocated as follows:

- 14 marks for algorithms, design choices and answer to the Question
- 2 marks for coding style and comments
- 14 marks based on performance on the (unseen) test set

The performance mark will be based on the accuracy of your network performance on the testing dataset, which is disjoint from the validation dataset we use to test your model at submission time.

General Advice:

- Use the variable `train_val_split` to help you make design decisions aimed to avoid overfitting to the training data. At the very end, you may wish to re-train using the entire training set.
- Try to be methodical in your development. Blindly modifying code, looking at the output, then modifying again can cause you to go around in circles. A better approach is to keep a record of what you have tried, and what outcome you observed. Decide on a hypothesis you want to test, run an experiment and record the result. Then move on to the next idea.

- You should consider the submission test script to be the final arbiter with regard to whether a certain approach is valid. If you try something, and the submission test runs and you get a good accuracy then the approach is valid. If it causes errors then it is not valid.
- Do Not leave this assignment to the last minute. Get started early, and submit early in order to ensure your code runs correctly. Marks from automated testing are final. You should aim to be uploading your final submission at least two hours before the deadline. It is likely that close to the deadline, the wait time on submission test results will increase.

Common Questions:

- **Can I train on the full dataset if I find it?** No. You should NOT attempt to reconstruct the test set by searching the Internet. We will retrain a random selection of submissions, as well as those achieving high accuracy. If your code attempts to search or load external assets, or we find a mismatch between your submittied code and saved model, you will receive zero marks.

- **My model is only slightly larger than 50MB, can you still accept it?** No, the 50MB limit is part of the assignment specification and is quite generous. You should be able to get away with much less.

- **Can we assume you will call `net.eval()` on our model prior to testing?** Yes.

Plagiarism Policy

Your program must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (as well as submissions to similar assignments from previous terms, if appropriate) and serious penalties will be applied, particularly in the case of repeat offences.

DO NOT COPY FROM OTHERS; DO NOT ALLOW ANYONE TO SEE YOUR CODE

Please refer to the UNSW Policy on Academic Integrity and Plagiarism if you require further clarification on this matter.

Good luck!
