CSCM45J Big Data and Machine Learning Coursework: Object Recognition

Policy

1. To be completed by students working individually.

2. **Feedback:** Individual feedback is given directly in the viva assessment. Feedback on the report is given via the rubric within Canvas.

3. **Learning outcome:** The tasks in this assignment are based on both your practical work in the lab sessions and your understanding of the theories and methods. Thus, through this coursework, you are expected to demonstrate both practical skills and theoretical knowledge that you have learned through this module. You also learn to formally present your understandings through technical writing. It is an opportunity to apply analytical and critical thinking, as well as practical implementation.

4. **Unfair practice:** This work is to be attempted individually. You may get help from your lecturer, academic tutor, and lab tutor, but you may not collaborate with your peers. **Copy and paste from the internet is not allowed**. **Using external code without proper referencing is also considered as breaching academic integrity.**

5. **University Academic Integrity and Academic Misconduct Statement:** By submitting this coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct.

   The policy can be found at https://www.swansea.ac.uk/academic-services/academic-guide/assessment-issues/academic-integrity-academic-misconduct.

6. **Submission deadline:** Both the report **and** your implemented code in Python need to be submitted electronically to Canvas by **11AM Monday 19th April**.

7. **Viva:** Viva will take place in the lab sessions following the submission deadline. You will be asked to select a timeslot for your viva in advance of the submission deadline. In your viva you will be asked questions about the reasoning behind your methodological choices, and to explain your findings.

1. Task

The amount of image data is growing exponentially, due in part to convenient and cheap camera equipment. Teaching computers to recognise objects within a scene has tremendous application prospects, with applications ranging from medical diagnostics to Snapchat filters. Object recognition problems have been studied for years in machine learning and computer vision fields; however, it is still a challenging and open problem for both academic and industry researchers. The following task is hopefully your first small step on this interesting question within machine learning.

You are provided with a small image dataset, where there are 100 different categories of objects, each of which has 500 images for training and 100 images for testing. Each individual image only contains one object. The task is to apply machine learning algorithms to classify the testing images into object categories. Code to compute image features and visualize an image is provided. You can use it to visualize the images and compute features to use in your machine learning algorithms. You will then use a model to perform classification and report quantitative results. You do not have to use all the provided code or methods discussed in the labs so far. You may add additional steps to the process if you wish. You are encouraged to use the implemented methodology from established Python packages taught in the labsheets (i.e. sklearn, skimage, keras, scipy…).

2. Image Dataset – Subset of CIFAR-100

We provide the 100 object categories from the complete CIFAR-100 dataset. Each category contains 500 training images and 100 testing images, which are stored in two 4D arrays. The corresponding category labels are also provided. The objects are also grouped into 20 "super-classes". The size of each image is fixed at 32x32x3, corresponding to height, width, and colour channel, respectively. The training images will be used to train your model(s), and the testing images will be used to evaluate your model(s). You can download the image dataset and relevant code for visualization and feature extraction from the Canvas page.

There are **six** numpy files provided, as follows:

- *trnImage*, 32x32x3x50000 matrix, training images (RGB image)
- *trnLabel_fine,* 50000x1 matrix, training labels (fine granularity)
- *trnLabel_coarse, 5*0000x1 matrix, training labels (coarse granularity)
- *tstImage*, 32x32x3x10000 matrix, testing images (RGB image)
- *tstLabel_fine,* 10000x1 matrix, testing labels (fine granularity)
- *tstLabel_coarse,* 10000x1 matrix, testing labels (coarse granularity)

The data is stored within a 4D matrix, and for many of you this will be the first time seeing a high dimensionality tensor. Although this can seem intimidating, it is relatively straightforward. The first dimension is the height of the image, the second dimension is the width, the third dimension is the colour channels (RGB), and the fourth dimension is the samples. Indexing into the matrix is like as with any other numeric array in Python, but now we deal with the additional dimensions. So, in a 4D matrix 'X', to index all pixels in all channels of the 5^{th} image, we use the index notation X[:, :, :, 4]. So, in a generic form, if we want to index into the i,j,k,l^{th} element of X we use X[i, j, k, l].

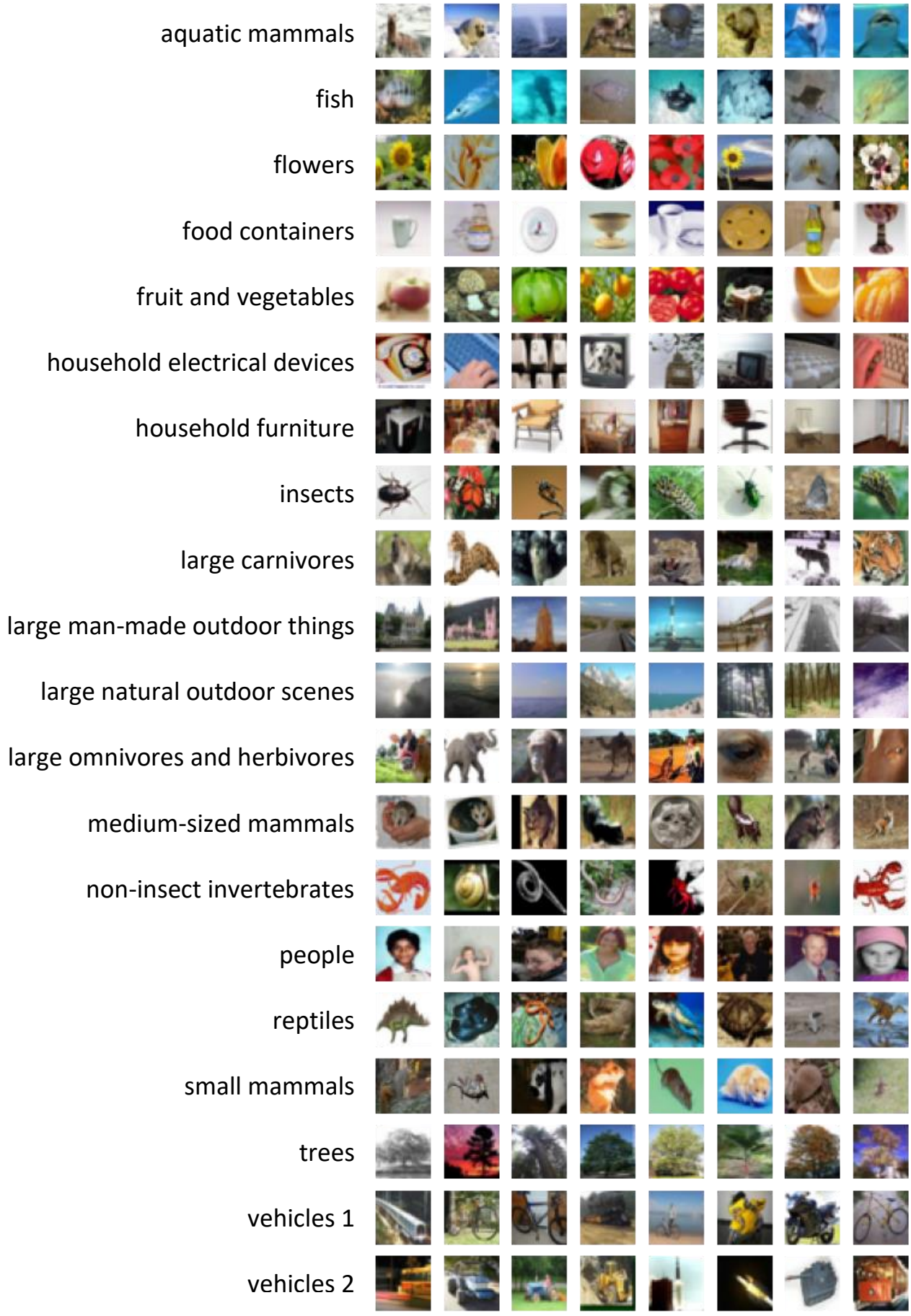

Figure 1. Coarse Categories of CIFAR-100 Dataset

3. Computing Features and Visualizing Images

A notebook, RunMe.ipynb, is provided to explain the concept of computing image features. The notebook is provided to showcase how to use the **skimage.feature.hog()** function to obtain features we wish to train our models on, how to visualize these features as an image, and how to visualize a raw image from the 4D array.

The function utilises the Histogram of Orientated Gradients method to represent image domain features as a vector. You are NOT asked to understand how these features are extracted from the images, but feel free to explore the algorithm, underlying code, and the respective Python package APIs. You can simply treat the features as the same as the features you loaded from Fisher Iris dataset in the Lab work.

4. Learning Algorithms

You can find all relative learning algorithms in the lab sheets and lecture notes. You can use the following algorithms (Python (and associated packages) built-in functions) to analyse the data and carry out the classification task. Please note: if you feed certain algorithms with a large chunk of data, it may take a long time to train. Not all methods are relevant to the task.

- Lab sheet 2:
    - K-Means
    - Gaussian Mixture Models
- Lab sheet 3:
    - Linear Regression
    - Principal Component Analysis
    - Linear Discriminative Analysis
- Lab sheet 4:
    - Support Vector Machine
    - Neural Networks
    - Convolutional Neural Networks

5. Benchmark and Discussion

Your proposed method should be trained on the training set alone, and then evaluated on the testing set. To evaluate: you should count, for each category, the percentage of correct recognition (i.e., classification), and report the confusion matrix.

The benchmark to compare against your methods with is 39.43%, averaged across all 20 super categories, and 24.49% for the finer granularity categories. Note: this is a reference, not a target.

6. Assessment

You are required to write a 4-page report to summarize your proposed method and the results. Your report should contain the following sections:

1. **Introduction**. Provide overview of the problem, the proposed solution, and your experimental results.
2. **Method.** Present your proposed method in detail. This should cover how the features are extracted, any feature processing you use (e.g., clustering and histogram generation, dimensionality reduction), which classifier(s) is/are used, and how they are trained and tested. This section may contain multiple sub-sections.
3. **Results.** Present your experimental results in this section. Explain the evaluation metric(s) you use and present the quantitative results (including the confusion matrix).
4. **Conclusion.** Provide a summary for your method and the results. Provide your critical analysis; including shortcomings of the methods and how they may be improved.
5. **References.** Include correctly formatted references where appropriate. References are not included in the page limit.

Page Limit: The report should be no more than **4 pages**. Font size should be no smaller than 10, and the text area is approximately 9.5x6 inches. You may use images but do so with care; do not use images to fill up the pages. You may use an additional cover sheet, which has your name and student number.

Source Code: Your submission should be professionally implemented and must be formatted as a Jupyter notebook. You may produce your notebook either locally within Jupyter, or you may utilize Google Colab to develop your notebook, however your submission **must** be a .ipynb notebook. Remember to carefully structure, comment, and markdown your implementation for clarity.

7. Submission

The assignment will be assessed via a viva in the lab sessions following the submission deadline. The markers will ask you questions about your approach and findings, and you should show them your code and report as supporting evidence during the viva. You will be given the marking rubric in advance of the submission deadline. This assignment is worth 20% of the total module credit.

Submit your work electronically to Canvas. **Your report should be in PDF format only**. Submit the code, together with your report, in a single Zip file with the filename of 123456.zip, where 123456 is your student number. The deadline for this coursework is **11AM Monday 19th April**.