# CSC345B/M45J Big Data and Machine Learning Coursework: Land Cover Detection using Aerial Imagery

## Policy

1. To be completed by students working individually.
2. Feedback: Individual 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/academicguide/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 **Friday 12 May (11:00 AM)**.

## 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 in medical diagnostics, remote sensing, traffic surveillance, Snapchat filters, etc. Automatic object detection has 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 about land cover detection using remotely sensed images as one of the examples of object detection for remote sensing applications. Hopefully, it will be your first small step on the interesting object detection problem within machine learning domain.

You are provided with a small image dataset. There are 21 different categories of landcover, each has 100 images. Each individual image mainly contains one type of land cover. The task is to first shuffle the data set samples and divide them into 80% train, 10% validation and 10% test samples. Then, apply machine learning algorithms to classify the train, validation and

testing images into object categories. The code to compute some 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 can also use convolutional deep learning modelling strategy for automatic feature extraction and classification.

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 lab sheets (i.e. sklearn, skimage, keras, scipy…). You must present a scientific approach, where you make suitable **comparison between at least two methods**.

## 2. Image Dataset – UC Merced Land Use Dataset

For this project, 21 land cover categories from the UC Merced Land Use dataset are provided. Originally, the images were manually extracted from large images of the USGS National Map Urban Area Imagery collection for various urban areas around US. The pixel resolution of this public domain imagery is 1 foot. Each category contains a total of 100 RGB colour images in two different resolutions, $242 \times 242 \times 3$ and $61 \times 61 \times 3$. These numbers correspond to height, width, and colour channel of each image, respectively. Then, there is a 4D array $X_{HighRes} = 2100 \times 242 \times 242 \times 3$ including high resolution images and a 4D array $X_{LowRes} = 2100 \times 61 \times 61 \times 3$ including low resolution images. In addition, the corresponding vector of labels $Y$ of size $2100 \times 1$ is provided.

You need to shuffle data and divide it into 80% train images as well as 10% validation and 10% test images for your implementations. The train and validation data will be used to develop your model(s) and optimize them, and the test images will be used to evaluate your model(s).

There are three numpy files provided, as follows:

- **DS_Xdata.npy**, including the array $X_{LowRes} = 2100 \times 61 \times 61 \times 3$ of 2100 low resolution RGB images of size $61 \times 61 \times 3$.
- **Xdata.npy**, including the array $X_{HighRes} = 2100 \times 242 \times 242 \times 3$ of 2100 high resolution RGB images of size $242 \times 242 \times 3$
- **Ydata.npy** including the array of labels ( $Y_{2100\times1}$) corresponding to the images in $X_{LowRes}$ and $X_{HighRes}$. The labels are sorted based on class numbers so that, the first 100 labels all belong to the first class and similarly it is the case for other classes.

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 samples, the second dimension is the height of the image, the third dimension is the width, the fourth dimension is the colour channels (RGB). 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 5th image, we use the index notation $X[4, :, :, :]$. Then, in a generic form, if we want to index into the $i, j, k, l^{th}$ element of $X$ we use $X[i, j, k, l]$.

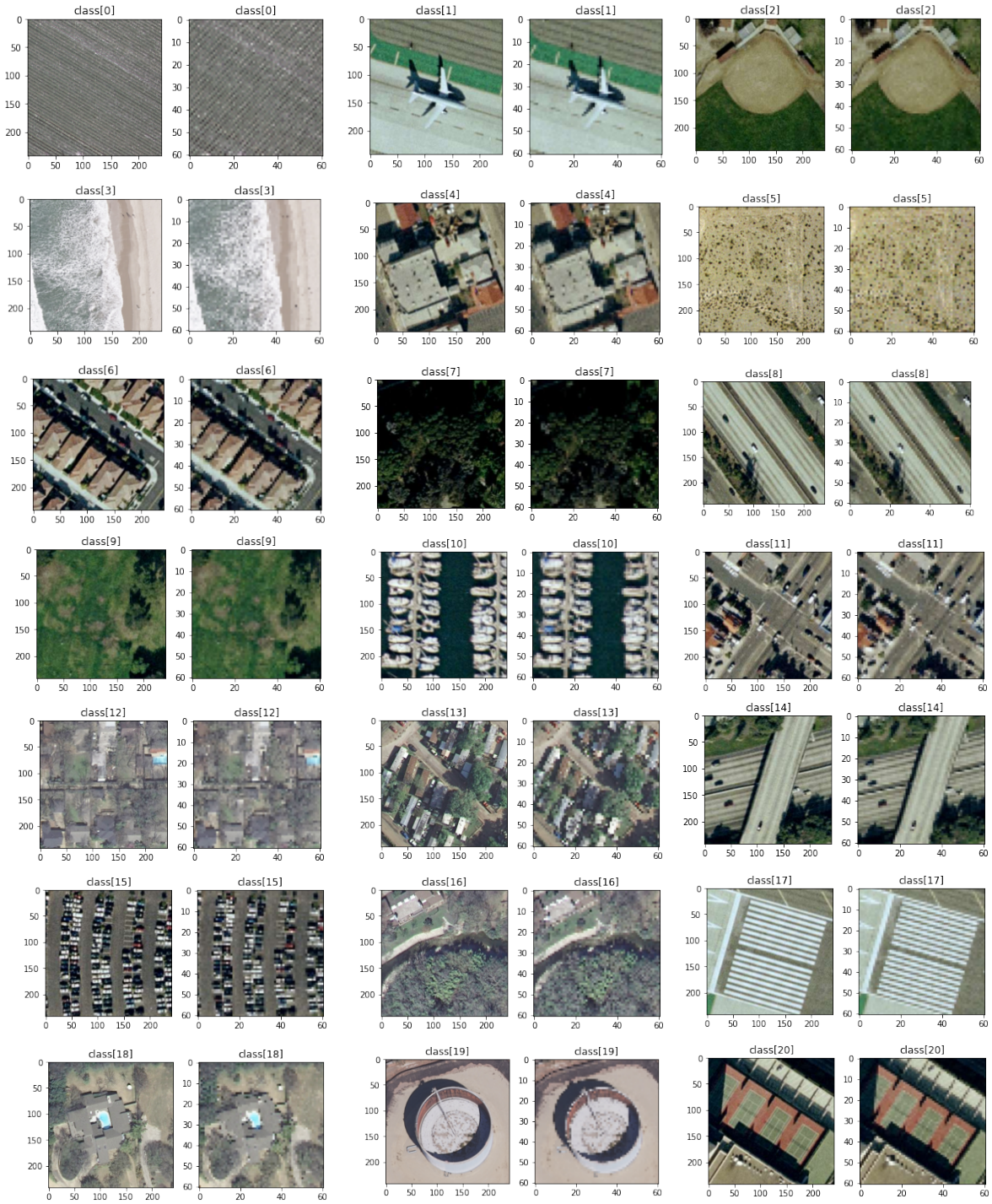


Figure 1. The high-resolution and low-resolution images of 21 classes from the UC Merced data set are visualized next to each other.

Figure 1 below shows images in both high-resolution and low-resolution images for all 21 classes. The name of classes in order are: ***agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbour, intersection, medium residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, tennis court***.

## 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 some features that can be used to train classification 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. Note that the **hog()** method can return two outputs, the first are the features, the second is an image representation of those features. Computing the second output is costly and not needed, but RunMe.ipynb provides it for your information.

## 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

You can **choose one of the high-resolution** or **low-resolution image sets** for your analysis. Using the low-resolution images will reduce the training time**. If you do not have access to a powerful machine, don't use high-resolution images** since the analysis will be very time consuming. In your analysis, you can choose **one of the following two strategies** for selection of data and methods:

**1.** Use all images including all the 21 classes and compare the results of two classification methods.

**2.** Use **at least one method on all images (21 classes)**. Then, select images from a **minimum of 4 classes, apply two different classification methods and compare their results.**

Your proposed method should be trained on the training set alone, and then be 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. Note that the confusion matrix can be large, and so you may need to think of ways to present appropriately. Please report the results for all three sets of train, validation, and test in terms of accuracy and visualise the confusion matrices for test set. Please tune the models in terms of their parameters, to avoid over-fitting or under-fitting and describe that in your report. If you extracted lots of features and the models suffer from over-fitting you might consider dimension reduction methods. In addition, you might be interested to use the model selection and some other model evaluation methods introduced in this module (You won't lose mark if not reported). You can download the image dataset and the relevant code for visualization and feature extraction from the Canvas page. Your report will contain a section in which you discuss your results.

## 6. Assessment

You are required to write a **4-page conference/publication style** 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.

6. **Appendices.** You may include appendix content if you wish for completeness, however the content you want graded must be in the main body of the report. Appendices 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 should 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 an **ipynb notebook**. You may produce your notebook either locally (Jupyter, VSCode etc.),

or you may utilize Google Colab to develop your notebook, however your submission must be an ipynb notebook. Remember to carefully structure, **comment, and markdown** your implementation for clarity.

## 7. Submission

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**.

Your code must be in a .ipynb format. Both **files should be named with your student number, i.e. 123456.pdf and 123456.ipynb**, where 123456 is your student number**. Do not zip your files** together.

The deadline for this coursework is **Friday 12 May (11:00 AM)**.